

The sTeX3 Package *

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

2022-02-09

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 2022-02-09)

Contents

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

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

21	document-structure.sty: 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	55
21.2.4	Structure Sharing	56
21.2.5	Global Variables	56
21.2.6	Colors	57
21.3	Limitations	57
22	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	61
22.3	Limitations	61
23	problem.sty: An Infrastructure for formatting Problems	62
23.1	Introduction	62
23.2	The User Interface	62
23.2.1	Package Options	62
23.2.2	Problems and Solutions	63
23.2.3	Multiple Choice Blocks	64
23.2.4	Including Problems	64
23.2.5	Reporting Metadata	64
23.3	Limitations	64
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	66
24.1	Introduction	67
24.2	The User Interface	67
24.2.1	Package and Class Options	67
24.2.2	Assignments	67
24.2.3	Typesetting Exams	67
24.2.4	Including Assignments	68
24.3	Limitations	68
IV	Implementation	70

25	STeX-Basics Implementation	71
25.1	The STeXDocument Class	71
25.2	Preliminaries	71
25.3	Messages and logging	72
25.4	Persistence	73
25.5	HTML Annotations	73
25.6	Languages	76
25.7	Activating/Deactivating Macros	77
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
27	STeX-References Implementation	90
27.1	Document URIs and URLs	90
27.2	Setting Reference Targets	92
27.3	Using References	93
28	STeX-Modules Implementation	95
28.1	The module environment	98
28.2	Invoking modules	104
29	STeX-Module Inheritance Implementation	106
29.1	SMS Mode	106
29.2	Inheritance	110
30	STeX-Symbols Implementation	115
30.1	Symbol Declarations	115
30.2	Notations	121
31	STeX-Terms Implementation	131
31.1	Symbol Invocations	131
31.2	Terms	134
31.3	Notation Components	140
32	STeX-Structural Features Implementation	143
32.1	Imports with modification	143
32.2	The feature environment	149
32.3	Features	151
33	STeX-Statements Implementation	157
33.1	Definitions	157
33.2	Assertions	160
33.3	Examples	162
33.4	Logical Paragraphs	164

34 The Implementation	167
34.1 Package Options	167
34.2 Proofs	167
34.3 Justifications	173
35 \LaTeX-Others Implementation	175
36 \LaTeX-Metatheory Implementation	176
37 Tikzinput Implementation	179
38 document-structure.sty Implementation	181
38.1 The OMDoc Class	181
38.2 Class Options	181
38.3 Beefing up the <code>document</code> environment	182
38.4 Implementation: OMDoc Package	182
38.5 Package Options	182
38.6 Document Structure	184
38.7 Front and Backmatter	187
38.8 Global Variables	189
39 MiKoSlides – Implementation	190
39.1 Class and Package Options	190
39.2 Notes and Slides	192
39.3 Header and Footer Lines	196
39.4 Frame Images	197
39.5 Colors and Highlighting	198
39.6 Sectioning	199
39.7 Excursions	201
40 The Implementation	203
40.1 Package Options	203
40.2 Problems and Solutions	204
40.3 Multiple Choice Blocks	209
40.4 Including Problems	210
40.5 Reporting Metadata	211
41 Implementation: The hwexam Class	213
41.1 Class Options	213
42 Implementation: The hwexam Package	215
42.1 Package Options	215
42.2 Assignments	216
42.3 Including Assignments	219
42.4 Typesetting Exams	220
42.5 Leftovers	222

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

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

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using Semantic Macros

TODO

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

1. If the TeX package is loaded with the option `mathhub=/path/to/mathhub`, then TeX will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the TeX-package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise, TeX will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smglob/calculus
narration-base: http://mathhub.info/smglob/calculus
dependencies: smglom/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

`url`: 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

5.1 Advanced Structuring Mechanisms

Given modules:

Example 1

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

Module 5.1.1[magma]

Module 5.1.2[monoid]

Module 5.1.3[group]

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 2

```
\begin{module}{ring}
\begin{copymodule}{group}{addition}
\renamedcl[name=universe]{universe}{runiverse}
\renamedcl[name=plus]{operation}{rplus}
\renamedcl[name=zero]{unit}{rzero}
\renamedcl[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}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\compl}

Test: $\rtimes a{\rm rplus} c{\rm rtimes} de$$
\end{module}
```

```
Module 5.1.4[ring]
Test:  $a \cdot (c + d \cdot e)$ 
```

TODO: explain donotclone

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`
`\stex`

Both print this \TeX logo.

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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 3

```
\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 4

```
\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 5

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

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

.

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

Example 6

```
\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 7

```
\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 8

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

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

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

Example 9

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

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

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

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

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

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

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

Other Argument Types

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

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

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

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

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

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 10

```
\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 11

```
\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. $\int \int \int f(x,y,z) dx dy dz$?

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 as operator precedence should be smaller than B ’s argument precedences.

For example:

Example 12

```
\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

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

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

showmods ($\langle boolean \rangle$) Shows explicit module information at the document margins.

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

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

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

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

9.1 Macros and Environments

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

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

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

<code>\stex_add_to_sms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
---------------------------------	--

<code>\if@latexml</code>	L ^A T _E X2e and L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

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

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

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

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

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

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

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

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

Chapter 10

sTEX-MathHub

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

10.1 Macros and Environments

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

10.1.1 Files, Paths, URIs

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

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

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

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

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:N</code>	\underline{NTF} \star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

10.1.2 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

```
\stex_add_to_current_module:n
\STEXexport
```

Adds the provided tokens to the `content` field of the current module.

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

Namespace 1:
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

12.1.1 The module-environment

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

`\stex_module_setup:nn` `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`
 Sets up a new module with name $\langle name \rangle$ and optional parameters $\langle params \rangle$. In particular, sets `\l_stex_current_module_str` appropriately.

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

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

Test 4

```

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

```

```

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

```

Test 5

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

```
Module 12.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

`\STEXModule` `\STEXModule {⟨fragment⟩}`

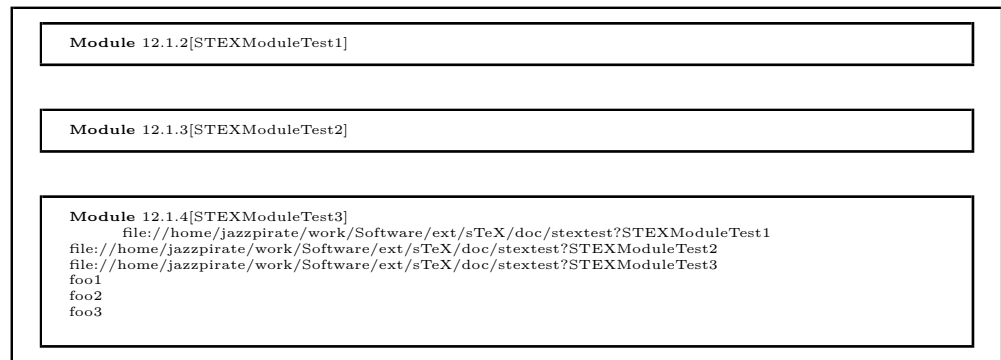
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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



`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists:

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

Module 13.1.1[Foo]
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

Meaning: `\macro->\protect \bar <`

Module 13.1.2[Importtest]
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

Module 13.1.3[Importtest2]
Meaning: `\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<`

`\usemodule` `\importmodule[⟨archive-ID⟩]{⟨module-path⟩}`

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

Test 9

```

\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:~\present\foo\\
\end{module}
\begin{module}{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{module}

```

Module 13.1.4[UseTest1]

Module 13.1.5[UseTest2]

Meaning: `>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo}<`

Module 13.1.6[UseTest3]

Meaning: `>undefined<`

Meaning: `>macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<`

All modules: `http://mathhub.info/sTeX?Metatheory`, `file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3`,
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2`
All symbols: `http://mathhub.info/sTeX?Metatheory?isa`, `http://mathhub.info/sTeX?Metatheory?bind`, `http://mathhub.info/sTeX?Metatheory?collec`,
`http://mathhub.info/sTeX?Metatheory?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collec`,
`http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`http://mathhub.info/sTeX?Metatheory?aseqfromto`, `http://mathhub.info/sTeX?Metatheory?aseqfromtovia`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar`

Test 10

```

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

```

Circular dependencies:

Module 13.1.7[CircDep1]

`>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}<`

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {⟨archive-ID⟩} {⟨module-path⟩}`

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

If `⟨archive-ID⟩` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `⟨archive-ID⟩` is empty:

(a) If `⟨path⟩` is empty, then `⟨name⟩` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `⟨name⟩.⟨lang⟩.tex` must exist in the same folder, containing a module `⟨name⟩`. That module should have the same namespace as the current one.

(b) If `⟨path⟩` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

(b) If `⟨path⟩` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

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

`\stex_import_require_module:nnnn`

`{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

Checks whether a module with URI `⟨ns⟩?⟨name⟩` 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 11

```
\begin{module}{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{module}
```

```
Module 14.1.1[SymdeclTest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

```
\begin{module}{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 14.1.2[NotationTest]

`\symdef`

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

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

Test 13

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

Module 14.1.3[SymdefTest]
 $a + b + c$

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> .

<hr/> <hr/>	<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S I E X brackets (by default $($ and $)$), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \S I E X for automated bracketing (by default $($ and $)$) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.

Test 14

```

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

```

Module 15.1.1[MathTest1]
 $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 15

```

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

```

Module 15.1.2[MathTest2]
 $\langle a \mid [b;c;d:e,f]^g \rangle$ and $\langle a \mid [b;c]^g \rangle$ and $\langle a \mid [b]^c \rangle$
 $a + (b \cdot c)$ 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 16

```
\begin{module}{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{module}
```

```
Module 15.1.3[TextTest]
  some a and some b and also some c here.
  some a and some b and also some c here.
  bar
  or just some c
  bar
  or first b, then c, and finally a
```

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.

`\ellipses`

TODO

Chapter 16

TeX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

<code>subproof</code>	The <code>pfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>method</code>	
<code>spfcases</code>	The <code>pfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcase</code> environments that mark up the cases one by one.
<code>spfcase</code>	The content of a <code>pfcases</code> environment are a sequence of case proofs marked up in the <code>pfcase</code> environment, which takes an optional <code>KeyVal</code> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <code>pfcase</code> environment is the same as that of a <code>proof</code> , i.e. <code>steps</code> , <code>proofcomments</code> , and <code>pfcases</code> environments. <code>\spfcasesketch</code> is a variant of the <code>spfcase</code> environment that takes the same arguments, but instead of the <code>spfsteps</code> in the body uses a third argument for a proof sketch.
<code>\spfcasesketch</code>	
<code>sproofcomment</code>	The <code>proofcomment</code> environment is much like a <code>step</code> , 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 <code>\premise</code> .

18.2.5 Proof End Markers

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

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

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

18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

The default meta theory for an sTeX module. Contains symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions).

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

The `omdoc` package is part of the $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ collection, a version of $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ that allows to markup $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$. This includes a simple structure sharing mechanism for $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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 $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ sources, or after translation.

21.1 Introduction

$\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ is a version of $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ that allows to markup $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

The `omdoc` 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 $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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

21.2 The User Interface

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

21.2.1 Package and Class Options

The `omdoc` 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 `omdoc` package accepts the same except the first two.

21.2.2 Document Structure

document

\documentkeys

id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble². This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the \LaTeX ML transformation.

omgroup

id

creators

contributors

short

loadmodules

The structure of the document is given by the `omgroup` environment just like in `OMDOC`. In the \LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the `omgroup`. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

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

```

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 `omdoc` package provides a variant `blindomgroup`

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.

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.

21.2.3 Ignoring Inputs

`ignore`
`showignores`

The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option

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

is given to the `omdoc` 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`.

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.

`\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

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.

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.

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

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

21.2.6 Colors

For convenience, the `omdoc` 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

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 `mikoslides` 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 `mikoslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the \TeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` 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 `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

22.2.1 Package Options

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

- | | |
|---|--|
| <code>slides</code>
<code>notes</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see 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 `mikoslides` 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 `notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

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.

⚠: 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 `mikoslides` 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 `nomtext` environment is just an `omtext` 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 `mikoslides` package is the \TeX logo it can be customized using `\setslidelogo{<logo name>}`.

The default footer line of the `mikoslides` 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 `mikoslides` 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}
```


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

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

`\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{nomtext}[title=Excursion]
                  \activateexcursion{founif}{../ex/founif}
                  We will cover first-order unification in \sref{founif}.
                  \end{nomtext}
```

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

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

```
\excursionref \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 [sTeXGitHub 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{problem}[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{problem}
\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).

```
Problem0.0 ()
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{problem}[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{problem}

```

Problem0.0 ()

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

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

Problem0.0 ()

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
`title` referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”,
`type` or “homework”), `given` (for the date the assignment was given), and `due` (for the date
`given` the assignment is due).
`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 \LaTeX source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical
`\testnewpage` space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage`
`\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 equiv-
`min` alent in number of minutes, and `reqpts` the points that are required for a perfect grade.
`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.

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

Example 8: A generated test heading.

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   showmods   .bool_set:N = \c_stex_showmods_bool ,
29   lang        .clist_set:N = \c_stex_languages_clist ,
30   mathhub     .tl_set_x:N = \mathhub ,
31   sms         .bool_set:N = \c_stex_persist_mode_bool ,
32   image       .bool_set:N = \c_tikzinput_image_bool ,
33   unknown     .code:n      = {}
34 }
35 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX

```

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

```

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

25.3 Messages and logging

```

43 <@@=stex_log>

Warnings and error messages
44 \msg_new:nnn{stex}{error/unknownlanguage}{
45   Unknown~language:~#1
46 }
47 \msg_new:nnn{stex}{warning/nomathhub}{
48   MATHHUB~system~variable~not~found~and~no~
49   \detokenize{\mathhub}~value~set!
50 }
51 \msg_new:nnn{stex}{error/deactivated-macro}{
52   The~\detokenize{#1}~command~is~only~allowed~in~#2!
53 }

```

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

```

54 \cs_new_protected:Nn \stex_debug:nn {
55   \clist_if_in:NnTF \c_stex_debug_clist { all } {
56     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
57       \\Debug~#1:~#2\\
58     }
59     \msg_none:nn{stex}{debug / #1}
60   }{
61     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
62       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
63         \\Debug~#1:~#2\\
64       }
65       \msg_none:nn{stex}{debug / #1}
66     }
67   }
68 }

```

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

Redirecting messages:

```

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

```

25.4 Persistence

78 `<@=stex_persist>`

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

```

79 \iow_new:N \c__stex_persist_sms_iow
80 \AddToHook{begindocument}{
81   \bool_if:NTF \c_stex_persist_mode_bool {
82     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
83   } {
84     % \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
85   }
86 }
87 \AddToHook{enddocument}{
88   \bool_if:NF \c_stex_persist_mode_bool {
89     % \iow_close:N \c__stex_persist_sms_iow
90   }
91 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

92 \cs_new_protected:Nn \stex_add_to_sms:n {
93   \bool_if:NF \c_stex_persist_mode_bool {
94     % \iow_now:Nn \c__stex_persist_sms_iow { #1 }
95   }
96 }

```

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

25.5 HTML Annotations

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

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

```

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

```

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

```

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

```

```

101 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
102 \fi
103
104 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
105   \if@latexml
106     \prg_return_true:
107   \else:
108     \prg_return_false:
109   \fi:
110 }

```

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

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

```

111 \tl_new:N \l__stex_annotate_arg_tl
112 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
113   \rustex_if:TF {
114     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
115   }{-}
116 }

```

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

`_stex_annotate_checkempty:n`

```

117 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
118   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
119   \tl_if_empty:NT \l__stex_annotate_arg_tl {
120     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
121   }
122 }

```

(End definition for `_stex_annotate_checkempty:n`.)

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

```

\stex_if_do_html:
123 \bool_new:N \l_stex_html_do_output_bool
124 \bool_set_true:N \l_stex_html_do_output_bool
125 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
126   \bool_if:nTF \l_stex_html_do_output_bool
127     \prg_return_true: \prg_return_false:
128 }

```

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

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

```

129 \cs_new_protected:Nn \stex_suppress_html:n {
130   \exp_args:Nne \use:nn {
131     \bool_set_false:N \l_stex_html_do_output_bool
132     #1
133   }{
134     \stex_if_do_html:T {
135       \bool_set_true:N \l_stex_html_do_output_bool
136     }
137   }
138 }

```


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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EXML, 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_EXML-implementations resort to perl bindings.

```

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

```

```

187   }{
188     \par\rustex_annotate_HTML_end:
189   }
190 }{
191   \latexml_if:TF {
192     \cs_new_protected:Nn \stex_annotate:nnn {
193       \__stex_annotate_checkempty:n { #3 }
194       \mode_if_math:TF {
195         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
196           \tl_use:N \l__stex_annotate_arg_tl
197         }
198       }{
199         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
200           \tl_use:N \l__stex_annotate_arg_tl
201         }
202       }
203     }
204     \cs_new_protected:Nn \stex_annotate_invisible:n {
205       \__stex_annotate_checkempty:n { #1 }
206       \mode_if_math:TF {
207         \cs:w latexml@invisible@math\cs_end:{
208           \tl_use:N \l__stex_annotate_arg_tl
209         }
210       } {
211         \cs:w latexml@invisible@text\cs_end:{
212           \tl_use:N \l__stex_annotate_arg_tl
213         }
214       }
215     }
216     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
217       \__stex_annotate_checkempty:n { #3 }
218       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
219         \tl_use:N \l__stex_annotate_arg_tl
220       }
221     }
222     \NewDocumentEnvironment{stex_annotate_env} { m m } {
223       \par\begin{latexml@annotateenv}{#1}{#2}
224     }{
225       \par\end{latexml@annotateenv}
226     }
227   }{
228     \cs_new_protected:Nn \stex_annotate:nnn {#3}
229     \cs_new_protected:Nn \stex_annotate_invisible:n {}
230     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
231     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
232   }
233 }

```

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

25.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

235 \prop_const_from_keyval:Nn \c_stex_languages_prop {
236   en = english ,
237   de = ngerman ,
238   ar = arabic ,
239   bg = bulgarian ,
240   ru = russian ,
241   fi = finnish ,
242   ro = romanian ,
243   tr = turkish ,
244   fr = french
245 }
246
247 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
248   english   = en ,
249   ngerman   = de ,
250   arabic    = ar ,
251   bulgarian = bg ,
252   russian   = ru ,
253   finnish   = fi ,
254   romanian  = ro ,
255   turkish   = tr ,
256   french    = fr
257 }
258 % todo: chinese simplified (zhs)
259 %       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:

```

260 \clist_if_empty:NF \c_stex_languages_clist {
261   \clist_clear:N \l_tmpa_clist
262   \clist_map_inline:Nn \c_stex_languages_clist {
263     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
264       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
265     } {
266       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
267     }
268   }
269   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
270   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
271 }

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

272 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
273   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
274   \def#1{
275     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
276   }
277 }

```

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

`\stex_reactivate_macro:N`

```
278 \cs_new_protected:Nn \stex_reactivate_macro:N {  
279   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname  
280 }
```

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

`\stex_do_aftergroup:n`

```
281 <@@=stex_aftergroup>  
282 \cs_new_protected:Nn \stex_do_aftergroup:n {  
283   \tl_gset:Nn \l__stex_aftergroup_tl { #1 }  
284   \aftergroup\__stex_aftergroup_do:  
285 }  
286 \cs_new_protected:Nn \__stex_aftergroup_do: {  
287   \l__stex_aftergroup_tl  
288 }
```

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

```
289 </package>
```

Chapter 26

STEX -MathHub Implementation

```
290 <*package>
291
292 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
293
294 <@@=stex_path>
295
296 Warnings and error messages
297 \msg_new:nnn{stex}{error/norepository}{
298   No~archive~#1~found~in~#2
299 }
300 \msg_new:nnn{stex}{error/notinarchive}{
301   Not~currently~in~an~archive,~but~\detokenize{#1}~
302   needs~one!
303 }
304 \msg_new:nnn{stex}{error/nofile}{
305   \detokenize{#1}~could~not~find~file~#2
306 }
```

26.1 Generic Path Handling

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

```
\stex_path_from_string:Nn
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
305 \cs_new_protected:Nn \stex_path_from_string:Nn {
306   \str_set:Nx \l_tmpa_str { #2 }
307   \str_if_empty:NTF \l_tmpa_str {
308     \seq_clear:N #1
309   }{
310     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
311     \sys_if_platform_windows:T{
312       \seq_clear:N \l_tmpa_tl
313       \seq_map_inline:Nn #1 {
314         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
315         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

316     }
317     \seq_set_eq:NN #1 \l_tmpa_tl
318   }
319   \stex_path_canonicalize:N #1
320 }
321 }
322 \cs_generate_variant:Nn \stex_path_from_string:Nn
323 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
324 \cs_new_protected:Nn \stex_path_to_string:NN {
325   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
326 }
327
328 \cs_new:Nn \stex_path_to_string:N {
329   \seq_use:Nn #1 /
330 }

```

(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
331 \str_const:Nn \c__stex_path_dot_str {.}
332 \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.

```

333 \cs_new_protected:Nn \stex_path_canonicalize:N {
334   \seq_if_empty:NF #1 {
335     \seq_clear:N \l_tmpa_seq
336     \seq_get_left:NN #1 \l_tmpa_tl
337     \str_if_empty:NT \l_tmpa_tl {
338       \seq_put_right:Nn \l_tmpa_seq {}
339     }
340     \seq_map_inline:Nn #1 {
341       \str_set:Nn \l_tmpa_tl { ##1 }
342       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
343         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
344           \seq_if_empty:NTF \l_tmpa_seq {
345             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
346               \c__stex_path_up_str
347             }
348           }{
349             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
350             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
351               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
352                 \c__stex_path_up_str
353               }
354             }{
355               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
356             }

```

```

357     }
358   }{
359     \str_if_empty:NF \l_tmpa_tl {
360       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
361     }
362   }
363 }
364 }
365 \seq_gset_eq:NN #1 \l_tmpa_seq
366 }
367 }

```

(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}`

```

368 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
369   \seq_if_empty:NTF #1 {
370     \prg_return_false:
371   }{
372     \seq_get_left:NN #1 \l_tmpa_tl
373     \str_if_empty:NTF \l_tmpa_tl {
374       \prg_return_true:
375     }{
376       \prg_return_false:
377     }
378   }
379 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

380 \str_new:N\l_stex_kpsewhich_return_str
381 \cs_new_protected:Nn \stex_kpsewhich:n {
382   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
383   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
384   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
385 }

```

(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`

```

386 \sys_if_platform_windows:TF{
387   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
388 }{
389   \stex_kpsewhich:n{-var-value~PWD}
390 }
391
392 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
393 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
394 \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

395 `<@@=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

396 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

397 `\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}`

398 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

399 `\c_stex_mainfile_str`

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

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

```
400 \seq_gclear_new:N\g_stex_currentfile_seq
401 \AddToHook{file/before}{
402   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
403   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
404     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
405   }{
406     \stex_path_from_string:Nn\g_stex_currentfile_seq{
407       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
408     }
409   }
410   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
411   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
412 }
413 \AddToHook{file/after}{
414   \seq_if_empty:NF\g__stex_files_stack{
415     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
416   }
417   \seq_if_empty:NTF\g__stex_files_stack{
418     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
419   }{
420     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
421     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
422   }
423 }
```

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

26.4 MathHub Repositories

```

424 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
425 \str_if_empty:NTF\mathhub{
426   \stex_kpsewhich:n{-var-value~MATHHUB}
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428
429   \str_if_empty:NTF\c_stex_mathhub_str{
430     \msg_warning:nn{stex}{warning/nomathhub}
431   }{
432     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
433     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434   }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
441   }
442   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(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
445 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
446   \str_set:Nx \l_tmpa_str { #1 }
447   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
448     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
449     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
450     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
451     \__stex_mathhub_find_manifest:N \l_tmpa_seq
452     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
453       \msg_error:nnxx{stex}{error/norepository}{#1}{
454         \stex_path_to_string:N \c_stex_mathhub_str
455       }
456     } {
457       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
458     }
459   }
460 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
461 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

462 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
463   \seq_set_eq:NN \l_tmpa_seq #1
464   \bool_set_true:N \l_tmpa_bool
465   \bool_while_do:Nn \l_tmpa_bool {
466     \seq_if_empty:NTF \l_tmpa_seq {
467       \bool_set_false:N \l_tmpa_bool
468     }{
469       \file_if_exist:nTF{
470         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
471       }{
472         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
473         \bool_set_false:N \l_tmpa_bool
474       }{
475         \file_if_exist:nTF{
476           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
477         }{
478           \seq_put_right:Nn \l_tmpa_seq{META-INF}
479           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
480           \bool_set_false:N \l_tmpa_bool
481         }{
482           \file_if_exist:nTF{
483             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
484           }{
485             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
486             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
487             \bool_set_false:N \l_tmpa_bool
488           }{
489             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
490           }
491         }
492       }
493     }
494   }
495   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
496 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

497 \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:

```

498 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
499   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
502     \str_set:Nn \l_tmpa_str {##1}
503     \exp_args:NNoo \seq_set_split:Nnn
504       \l_tmpb_seq \c_colon_str \l_tmpa_str
505     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

506 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508 }
509 \exp_args:No \str_case:nnTF \l_tmpa_tl {
510 {id} {
511 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512 { id } \l_tmpb_tl
513 }
514 {narration-base} {
515 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516 { narr } \l_tmpb_tl
517 }
518 {url-base} {
519 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520 { docurl } \l_tmpb_tl
521 }
522 {source-base} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { ns } \l_tmpb_tl
525 }
526 {ns} {
527 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528 { ns } \l_tmpb_tl
529 }
530 {dependencies} {
531 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532 { deps } \l_tmpb_tl
533 }
534 }{}{}
535 }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540 \stex_require_repository:n { #1 }
541 \prop_set_eq:Nc \l_stex_current_repository_prop {
542 c_stex_mathhub_#1_manifest_prop
543 }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547 \stex_debug:nn{mathhub}{Opening~archive:~#1}
548 \__stex_mathhub_do_manifest:n { #1 }
549 \exp_args:Nx \stex_add_to_sms:n {
550 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
551 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
552 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

553     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
554     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
555   }
556 }
557 }
558 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

559 %\prop_new:N \l_stex_current_repository_prop
560
561 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
562 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
563   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
564 } {
565   \__stex_mathhub_parse_manifest:n { main }
566   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
567   \l_tmpa_str
568   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
569   \c_stex_mathhub_main_manifest_prop
570   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
571   \stex_debug:nn{mathhub}{Current~repository:~
572   \prop_item:Nn \l_stex_current_repository_prop {id}
573 }
574 }

```

(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.

```

575 \cs_new_protected:Nn \stex_in_repository:nn {
576   \str_set:Nx \l_tmpa_str { #1 }
577   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
578   \str_if_empty:NTF \l_tmpa_str {
579     \prop_if_exist:NTF \l_stex_current_repository_prop {
580       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
581       \exp_args:Ne \l_tmpa_cs{
582         \prop_item:Nn \l_stex_current_repository_prop { id }
583       }
584     }{
585       \l_tmpa_cs{}
586     }
587   }{
588     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
589     \stex_require_repository:n \l_tmpa_str
590     \str_set:Nx \l_tmpa_str { #1 }
591     \exp_args:Nne \use:nn {
592       \stex_set_current_repository:n \l_tmpa_str
593       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
594     }{
595       \stex_debug:nn{mathhub}{switching~back~to:~
596       \prop_if_exist:NTF \l_stex_current_repository_prop {
597         \prop_item:Nn \l_stex_current_repository_prop { id }::~

```

```

598         \meaning\l_stex_current_repository_prop
599     }{
600         no~repository
601     }
602 }
603 \prop_if_exist:NTF \l_stex_current_repository_prop {
604     \stex_set_current_repository:n {
605         \prop_item:Nn \l_stex_current_repository_prop { id }
606     }
607 }{
608     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
609 }
610 }
611 }
612 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn
613 \newif \ifinputref \inputreffalse
614
615 \cs_new_protected:Nn \stex_mhinput:nn {
616     \stex_in_repository:nn {#1} {
617         \ifinputref
618             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
619         \else
620             \inputreftrue
621             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
622             \inputreffalse
623         \fi
624     }
625 }
626 \NewDocumentCommand \mhinput { 0{} m}{
627     \stex_mhinput:nn{ #1 }{ #2 }
628 }
629
630 \cs_new_protected:Nn \stex_inputref:nn {
631     \stex_in_repository:nn {#1} {
632         \bool_lazy_any:nTF {
633             {\rustex_if_p:} {\latexml_if_p:}
634         } {
635             \str_clear:N \l_tmpa_str
636             \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
637                 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
638             }
639             \stex_annotate_invisible:nnn{inputref}{
640                 \l_tmpa_str / #2
641             }{}
642         }{
643             \begingroup
644             \inputreftrue
645             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
646             \endgroup
647         }

```

```

648 }
649 }
650
651 \NewDocumentCommand \inputref { 0{ } m }{
652   \stex_inputref:nn{ #1 }{ #2 }
653 }
654
655 \cs_new_protected:Nn \stex_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }
659 }
660 \newcommand\addmhbibresource[2][]{
661   \stex_mhbibresource:nn{ #1 }{ #2 }
662 }

```

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

`\mhpath`

```

663 \def \mhpath #1 #2 {
664   \exp_args:Ne \str_if_eq:nnTF{#1}{-}{
665     \c_stex_mathhub_str /
666     \prop_item:Nn \l_stex_current_repository_prop { id }
667     / source / #2
668   }{
669     \c_stex_mathhub_str / #1 / source / #2
670   }
671 }

```

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

`\libinput`

```

672 \cs_new_protected:Npn \libinput #1 {
673   \prop_if_exist:NF \l_stex_current_repository_prop {
674     \msg_error:nnn{stex}{error/notinarchive}\libinput
675   }
676   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
677     \msg_error:nnn{stex}{error/notinarchive}\libinput
678   }
679   \bool_set_false:N \l_tmpa_bool
680   \tl_clear:N \l_tmpa_tl
681   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
682   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
683   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
684   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
685     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
686     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
687       / meta-inf / lib / #1.tex}{
688       \bool_set_true:N \l_tmpa_bool
689       \tl_put_right:Nx \l_tmpa_tl {
690         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
691           / meta-inf / lib / #1.tex}
692       }
693     }{}

```

```

694 }
695 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
696 / \l_tmpa_str / lib / #1.tex
697 }{
698   \bool_set_true:N \l_tmpa_bool
699   \tl_put_right:Nx \l_tmpa_tl {
700     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
701       / \l_tmpa_str / lib / #1.tex}
702   }
703 }{}
704 \bool_if:NF \l_tmpa_bool {
705   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
706 }
707 \l_tmpa_tl
708 }

```

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

```

709 </package>

```

Chapter 27

STEX -References Implementation

```
710 <*package>
711
712 %%%%%%%%%% references.dtx %%%%%%%%%%
713
714 %\RequirePackage{hyperref}
715 %\RequirePackage{cleveref}
716 <@@=stex_refs>
717
718 Warnings and error messages
719
720 \iow_new:N \c__stex_refs_refs_iow
721 \AddToHook{begindocument}{
722   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
723 }
724 \AddToHook{enddocument}{
725   \iow_close:N \c__stex_refs_refs_iow
726 }
727
728 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
729
730 \NewDocumentCommand \STEXreftitle { m } {
731   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
732 }
733
```

27.1 Document URIs and URLs

```
731 \seq_new:N \g__stex_refs_all_refs_seq
732
733 \str_new:N \l_stex_current_docns_str
734
735 \cs_new_protected:Nn \stex_get_document_uri: {
736   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
737   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
738   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
739   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
740 }
```



```

740 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
741
742 \str_clear:N \l_tmpa_str
743 \prop_if_exist:NT \l_stex_current_repository_prop {
744   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
745     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
746   }
747 }
748
749 \str_if_empty:NTF \l_tmpa_str {
750   \str_set:Nx \l_stex_current_docns_str {
751     file:/\stex_path_to_string:N \l_tmpa_seq
752   }
753 }{
754   \bool_set_true:N \l_tmpa_bool
755   \bool_while_do:Nn \l_tmpa_bool {
756     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
757     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
758       {source} { \bool_set_false:N \l_tmpa_bool }
759     }{}{
760       \seq_if_empty:NT \l_tmpa_seq {
761         \bool_set_false:N \l_tmpa_bool
762       }
763     }
764   }
765
766   \seq_if_empty:NTF \l_tmpa_seq {
767     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
768   }{
769     \str_set:Nx \l_stex_current_docns_str {
770       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
771     }
772   }
773 }
774 }
775
776 \str_new:N \l_stex_current_docurl_str
777 \cs_new_protected:Nn \stex_get_document_url: {
778   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
779   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
780   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
781   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
782   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
783
784   \str_clear:N \l_tmpa_str
785   \prop_if_exist:NT \l_stex_current_repository_prop {
786     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
787       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
788         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
789       }
790     }
791   }
792
793   \str_if_empty:NTF \l_tmpa_str {
794     \str_set:Nx \l_stex_current_docurl_str {

```

```

794     file:/\stex_path_to_string:N \l_tmpa_seq
795   }
796 }{
797   \bool_set_true:N \l_tmpa_bool
798   \bool_while_do:Nn \l_tmpa_bool {
799     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
800     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
801       {source} { \bool_set_false:N \l_tmpa_bool }
802     }{}{
803       \seq_if_empty:NT \l_tmpa_seq {
804         \bool_set_false:N \l_tmpa_bool
805       }
806     }
807   }
808
809   \seq_if_empty:NTF \l_tmpa_seq {
810     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
811   }{
812     \str_set:Nx \l_stex_current_docurl_str {
813       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
814     }
815   }
816 }
817 }

```

27.2 Setting Reference Targets

```

818 \str_const:Nn \c__stex_refs_url_str{URL}
819 \str_const:Nn \c__stex_refs_ref_str{REF}
820 % @currentlabel -> number
821 % @currentlabelname -> title
822 % @currentHref -> name.number <- id of some kind
823 % \theH# -> \arabic{section}
824 % \the# -> number
825 % \hyper@makecurrent{#}
826 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
827   \stex_get_document_uri:
828   \str_set:Nx \l_tmpa_str { #1 }
829   \str_if_empty:NT \l_tmpa_str {
830     \int_zero:N \l_tmpa_int
831     \bool_set_true:N \l_tmpa_bool
832     \bool_while_do:Nn \l_tmpa_bool {
833       \cs_if_exist:cTF {
834         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
835       }{
836         \int_incr:N \l_tmpa_int
837       }{
838         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843   \str_set:Nx \l_tmpa_str {
844     \l_stex_current_docns_str??\l_tmpa_str

```

```

845 }
846 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
847 \stex_if_smsmode:TF {
848   \stex_get_document_url:
849   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
850   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
851 }{
852   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
853   \exp_args:Nx\label{sref_\l_tmpa_str}
854
855   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
856   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
857 }
858 }
859 \cs_new_protected:Npn \stexauxadddocref #1 {
860   \str_set:Nx \l_tmpa_str {#1}
861   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
862   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
863 }
864 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
865   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
866 }

```

27.3 Using References

```

867 \str_new:N \l__stex_refs_indocument_str
868 \keys_define:nn { stex / sref } {
869   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
870   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
871   pre           .tl_set:N = \l__stex_refs_pre_tl ,
872   post          .tl_set:N = \l__stex_refs_post_tl ,
873   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
874 }
875
876 \bool_new:N \c__stex_refs_hyperref_bool
877 \bool_set_false:N \c__stex_refs_hyperref_bool
878 \AddToHook{begindocument}{
879   \ifpackageloaded{hyperref}{
880     \bool_set_true:N \c__stex_refs_hyperref_bool
881   }{}
882 }
883
884
885 \cs_new_protected:Nn \__stex_refs_args:n {
886   \tl_clear:N \l__stex_refs_linktext_tl
887   \tl_clear:N \l__stex_refs_fallback_tl
888   \tl_clear:N \l__stex_refs_pre_tl
889   \tl_clear:N \l__stex_refs_post_tl
890   \str_clear:N \l__stex_refs_repo_str
891   \keys_set:nn { stex / sref } { #1 }
892 }
893
894 \NewDocumentCommand \sref { 0{} m}{
895   \__stex_refs_args:n { #1 }

```

```

896 \str_if_empty:NTF \l__stex_refs_indocument_str {
897   \str_set:Nn \l_tmpa_str { #2 }
898   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
899   \tl_set:Nn \l_tmpa_tl {
900     \l__stex_refs_fallback_tl
901   }
902   \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
903     \str_set:Nn \l_tmpb_str { ##1 }
904     \str_if_eq:eeT { \l_tmpa_str } {
905       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
906     } {
907       \seq_map_break:n {
908         \tl_set:Nn \l_tmpa_tl {
909           % doc uri in \l_tmpb_str
910           \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
911           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
912             % reference
913             \cs_if_exist:cTF{autoref}{
914               \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
915             }{
916               \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
917             }
918           }{
919             % URL
920             \if_bool:N \c__stex_refs_hyperref_bool {
921               \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
922             }{
923               \l__stex_refs_fallback_tl
924             }
925           }
926         }
927       }
928     }
929   }
930   \l_tmpa_tl
931 }{
932   % TODO
933 }
934 }
935
936 \end{package}

```

Chapter 28

STEX -Modules Implementation

```
937 <*package>
938
939 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
940
941 <@@=stex_modules>
    Warnings and error messages
942 \msg_new:nnn{stex}{error/unknownmodule}{
943   No~module~#1~found
944 }
945 \msg_new:nnn{stex}{error/syntax}{
946   Syntax~error:~#1
947 }
948 \msg_new:nnn{stex}{error/siglanguage}{
949   Module~#1~declares~signature~#2,~but~does~not~
950   declare~its~language
951 }
952
953 \msg_new:nnn{stex}{error/concllictingmodules}{
954   Conflicting~imports~for~module~#1
955 }

\l_stex_current_module_str The current module:
956 \str_new:N \l_stex_current_module_str
    (End definition for \l_stex_current_module_str. This variable is documented on page 26.)

\l_stex_all_modules_seq Stores all available modules
957 \seq_new:N \l_stex_all_modules_seq
    (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)

\stex_if_in_module_p:
\stex_if_in_module:TF
958 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
959   \str_if_empty:NTF \l_stex_current_module_str
960     \prg_return_false: \prg_return_true:
961 }
```

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

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

```
962 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
963   \prop_if_exist:cTF { c_stex_module_#1_prop }
964   \prg_return_true: \prg_return_false:
965 }
```

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
966 \cs_new_protected:Nn \stex_add_to_current_module:n {
967   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
968 }
969 \cs_new_protected:Npn \STEXexport {
970   \begingroup
971   \newlinechar=-1\relax
972   \endlinechar=-1\relax
973   %\catcode'\ = 9\relax
974   \expandafter\endgroup\STEXexport:n
975 }
976 \cs_new_protected:Nn \STEXexport:n {
977   \ignorespaces #1
978   \stex_add_to_current_module:n { \ignorespaces #1 }
979   \stex_smsmode_set_codes:
980 }
981 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

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

`\stex_add_constant_to_current_module:n`

```
982 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
983   \str_set:Nx \l_tmpa_str { #1 }
984   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
985 }
986
987 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
988 % \str_set:Nx \l_tmpa_str { #1 }
989 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
990 %}
```

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

`\stex_collect_imports:n`

```
991 \cs_new_protected:Nn \stex_collect_imports:n {
992   \seq_clear:N \l_stex_collect_imports_seq
993   \__stex_modules_collect_imports:n {#1}
994 }
995 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
996   \seq_map_inline:cn {c_stex_module_#1_imports} {
997     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
998       \__stex_modules_collect_imports:n { ##1 }
999     }
1000 }
```

```

1000 }
1001 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1002   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
1003 }
1004 }

```

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

`\stex_add_import_to_current_module:n`

```

1005 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1006   \str_set:Nx \l_tmpa_str { #1 }
1007   \exp_args:Nno
1008   \seq_if_in:cnF{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str{
1009     \seq_gput_right:co{c_stex_module\_l_stex_current_module_str_imports}\l_tmpa_str
1010   }
1011 }

```

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

`\stex_modules_compute_namespace:nN`

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

```

1012 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1013   \str_set:Nx \l_tmpa_str { #1 }
1014   \seq_set_eq:NN \l_tmpa_seq #2
1015   % split off file extension
1016   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1017   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1018   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1019   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1020
1021   \bool_set_true:N \l_tmpa_bool
1022   \bool_while_do:Nn \l_tmpa_bool {
1023     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1024     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1025       {source} { \bool_set_false:N \l_tmpa_bool }
1026     }{}{
1027       \seq_if_empty:NT \l_tmpa_seq {
1028         \bool_set_false:N \l_tmpa_bool
1029       }
1030     }
1031   }
1032
1033   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1034   \str_if_empty:NTF \l_stex_modules_subpath_str {
1035     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1036   }{
1037     \str_set:Nx \l_stex_modules_ns_str {
1038       \l_tmpa_str/\l_stex_modules_subpath_str
1039     }
1040   }
1041 }

```

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

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1042 \str_new:N \l_stex_modules_ns_str
1043 \str_new:N \l_stex_modules_subpath_str

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

```

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

```

1044 \cs_new_protected:Nn \stex_modules_current_namespace: {
1045   \str_clear:N \l_stex_modules_subpath_str
1046   \prop_if_exist:NTF \l_stex_current_repository_prop {
1047     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1048     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1049   }{
1050     % split off file extension
1051     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1052     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1053     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1054     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1055     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1056     \str_set:Nx \l_stex_modules_ns_str {
1057       file:/\stex_path_to_string:N \l_tmpa_seq
1058     }
1059   }
1060 }

```

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

28.1 The module environment

module arguments:

```

1061 \keys_define:nn { stex / module } {
1062   title      .str_set_x:N = \l_stex_module_title_str ,
1063   ns         .str_set_x:N = \l_stex_module_ns_str ,
1064   lang       .str_set_x:N = \l_stex_module_lang_str ,
1065   sig        .str_set_x:N = \l_stex_module_sig_str ,
1066   creators   .str_set_x:N = \l_stex_module_creators_str ,
1067   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1068   meta       .str_set_x:N = \l_stex_module_meta_str ,
1069   srccite    .str_set_x:N = \l_stex_module_srccite_str
1070 }
1071
1072 \cs_new_protected:Nn \__stex_modules_args:n {
1073   \str_clear:N \l_stex_module_title_str
1074   \str_clear:N \l_stex_module_ns_str
1075   \str_clear:N \l_stex_module_lang_str
1076   \str_clear:N \l_stex_module_sig_str
1077   \str_clear:N \l_stex_module_creators_str
1078   \str_clear:N \l_stex_module_contributors_str
1079   \str_clear:N \l_stex_module_meta_str
1080   \str_clear:N \l_stex_module_srccite_str
1081   \keys_set:nn { stex / module } { #1 }

```



```

1082 }
1083
1084 % module parameters here? In the body?
1085

```

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

```

1086 \cs_new_protected:Nn \stex_module_setup:nn {
1087   \str_set:Nx \l_stex_module_name_str { #2 }
1088   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1089 \stex_if_in_module:TF {
1090   % Nested module
1091   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1092     { ns } \l_stex_module_ns_str
1093   \str_set:Nx \l_stex_module_name_str {
1094     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1095       { name } / \l_stex_module_name_str
1096   }
1097 }{
1098   % not nested:
1099   \str_if_empty:NT \l_stex_module_ns_str {
1100     \stex_modules_current_namespace:
1101     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1102     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1103       / {\l_stex_module_ns_str}
1104     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1105     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1106       \str_set:Nx \l_stex_module_ns_str {
1107         \stex_path_to_string:N \l_tmpa_seq
1108       }
1109     }
1110   }
1111 }

```

Next, we determine the language of the module:

```

1112 \str_if_empty:NT \l_stex_module_lang_str {
1113   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1114   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1115   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1116   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1117   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1118     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1119       inferred~from~file~name}
1120     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1121   }
1122 }
1123
1124 \str_if_empty:NF \l_stex_module_lang_str {
1125   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1126     \l_tmpa_str {
1127     \ltx@ifpackageloaded{babel}{
1128       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1129     }{}
1130   } {
1131     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1132   }
1133 }

```

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

```

1134 \str_if_empty:NTF \l_stex_module_sig_str {
1135   \exp_args:Nnx \prop_gset_from_keyval:cn {
1136     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1137   } {
1138     name      = \l_stex_module_name_str ,
1139     ns        = \l_stex_module_ns_str ,
1140     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1141     lang      = \l_stex_module_lang_str ,
1142     sig       = \l_stex_module_sig_str ,
1143     meta      = \l_stex_module_meta_str
1144   }
1145   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1146   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1147   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1148   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1149   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1150 \str_if_empty:NT \l_stex_module_meta_str {
1151   \str_set:Nx \l_stex_module_meta_str {
1152     \c_stex_metatheory_ns_str ? Metatheory
1153   }
1154 }
1155 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1156   \bool_set_true:N \l_stex_in_meta_bool
1157   \exp_args:Nx \stex_add_to_current_module:n {
1158     \bool_set_true:N \l_stex_in_meta_bool
1159     \stex_activate_module:n {\l_stex_module_meta_str}
1160     \bool_set_false:N \l_stex_in_meta_bool
1161   }
1162   \stex_activate_module:n {\l_stex_module_meta_str}
1163   \bool_set_false:N \l_stex_in_meta_bool
1164 }
1165 }{
1166   \str_if_empty:NT \l_stex_module_lang_str {
1167     \msg_error:nnxx{stex}{error/siglanguage}{
1168       \l_stex_module_ns_str?\l_stex_module_name_str
1169     }{\l_stex_module_sig_str}
1170   }
1171
1172   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1173   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1174   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1175   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1176   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1177   \str_set:Nx \l_tmpa_str {

```

```

1178     \stex_path_to_string:N \l_tmpa_seq /
1179     \l_tmpa_str . \l_stex_module_sig_str .tex
1180 }
1181 \IfFileExists \l_tmpa_str {
1182     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1183         \seq_clear:N \l_stex_all_modules_seq
1184         %\prop_clear:N \l_stex_current_module_prop
1185         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1186         \input { \l_tmpa_str }
1187     }
1188 }{
1189     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1190 }
1191 \stex_activate_module:n {
1192     \l_stex_module_ns_str ? \l_stex_module_name_str
1193 }
1194 %\prop_set_eq:Nc \l_stex_current_module_prop {
1195 %   c_stex_module_
1196 %   \l_stex_module_ns_str ?
1197 %   \l_stex_module_name_str
1198 %   _prop
1199 %}
1200 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1201 }
1202 }

```

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

module The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1203 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1204     \stex_reactivate_macro:N \STEXexport
1205     \stex_reactivate_macro:N \importmodule
1206     \stex_reactivate_macro:N \symdecl
1207     \stex_reactivate_macro:N \notation
1208     \stex_reactivate_macro:N \symdef
1209     \stex_module_setup:nn{#1}{#2}
1210
1211     \stex_debug:nn{modules}{
1212         New~module:\\
1213         Namespace:~\l_stex_module_ns_str\\
1214         Name:~\l_stex_module_name_str\\
1215         Language:~\l_stex_module_lang_str\\
1216         Signature:~\l_stex_module_sig_str\\
1217         Metatheory:~\l_stex_module_meta_str\\
1218         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1219     }
1220
1221     \seq_put_right:Nx \l_stex_all_modules_seq {
1222         \l_stex_module_ns_str ? \l_stex_module_name_str
1223     }
1224
1225     % \seq_gput_right:Nx \g_stex_modules_in_file_seq

```

```

1226 %      { \l_stex_module_ns_str ? \l_stex_module_name_str }
1227
1228 \stex_if_smsmode:TF {
1229   \stex_smsmode_set_codes:
1230 } {
1231   \begin{stex_annotate_env} {theory} {
1232     \l_stex_module_ns_str ? \l_stex_module_name_str
1233   }
1234
1235   \stex_annotate_invisible:nnn{header}{} {
1236     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1237     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1238     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1239       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1240     }
1241   }
1242 }
1243 % TODO: Inherit metatheory for nested modules?
1244 }
1245 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:nn`.)

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

```

1246 \cs_new_protected:Nn \_stex_modules_end_module: {
1247 %   \str_set:Nx \l_tmpa_str {
1248 %     c_stex_module_
1249 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1250 %     \prop_item:Nn \l_stex_current_module_prop { name }
1251 %     _prop
1252 %   }
1253 %^^A \prop_new:c { \l_tmpa_str }
1254 %   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1255   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1256 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1257 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1258 \NewDocumentEnvironment { @module } { 0 } { m } {
1259   \par
1260   \_stex_modules_begin_module:nn{#1}{#2}
1261 } {
1262   \_stex_modules_end_module:
1263   \stex_if_smsmode:TF {
1264 %     \exp_args:Nx \stex_add_to_sms:n {
1265 %       \prop_gset_from_keyval:cn {
1266 %         c_stex_module_
1267 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1268 %         \prop_item:Nn \l_stex_current_module_prop { name }
1269 %         _prop
1270 %       } {
1271 %         name          = \prop_item:cn { \l_tmpa_str } { name } ,

```

```

1272 %      ns      = \prop_item:cn { \l_tmpa_str } { ns } ,
1273 %      file     = \prop_item:cn { \l_tmpa_str } { file } ,
1274 %      lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
1275 %      sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
1276 %      meta     = \prop_item:cn { \l_tmpa_str } { meta }
1277 %    }
1278 %  }
1279 }{
1280   \end{stex_annotate_env}
1281 }
1282 }

```

\stex_modules_heading: Code for document headers

```

1283 \cs_if_exist:NTF \thesection {
1284   \newcounter{module}[section]
1285 }{
1286   \newcounter{module}
1287 }
1288
1289 \bool_if:NT \c_stex_showmods_bool {
1290   \latexml_if:F { \RequirePackage{mdframed} }
1291 }
1292
1293 \cs_new_protected:Nn \stex_modules_heading: {
1294   \stepcounter{module}
1295   \par
1296   \bool_if:NT \c_stex_showmods_bool {
1297     \noindent{\textbf{Module} ~
1298       \cs_if_exist:NT \thesection {\thesection.}
1299       \themodule ~ [\l_stex_module_name_str]
1300     }
1301     \str_if_empty:NTF \l_stex_module_title_str {
1302       }{
1303         \quad(\l_stex_module_title_str)\hfill
1304       }\par
1305     }
1306     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1307     % TODO
1308     \stex_ref_new_doc_target:n \l_stex_module_name_str
1309   }

```

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

Finally:

```

1310 \NewDocumentEnvironment { module } { 0{} m } {
1311   \bool_if:NT \c_stex_showmods_bool {
1312     \begin{mdframed}
1313   }
1314   \begin{@module}[#1]{#2}
1315   \stex_modules_heading:
1316 }{
1317   \end{@module}
1318   \bool_if:NT \c_stex_showmods_bool {
1319     \end{mdframed}
1320   }

```

```
1321 }
```

28.2 Invoking modules

```
\STEXModule
\stex_invoke_module:n 1322 \NewDocumentCommand \STEXModule { m } {
1323   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1324   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1325   \tl_set:Nn \l_tmpa_tl {
1326     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1327   }
1328   \seq_map_inline:Nn \l_stex_all_modules_seq {
1329     \str_set:Nn \l_tmpb_str { ##1 }
1330     \str_if_eq:eeT { \l_tmpa_str } {
1331       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1332     } {
1333       \seq_map_break:n {
1334         \tl_set:Nn \l_tmpa_tl {
1335           \stex_invoke_module:n { ##1 }
1336         }
1337       }
1338     }
1339   }
1340   \l_tmpa_tl
1341 }
1342
1343 \cs_new_protected:Nn \stex_invoke_module:n {
1344   \stex_debug:nn{modules}{Invoking~module~#1}
1345   \peek_charcode_remove:NTF ! {
1346     \__stex_modules_invoke_uri:nN { #1 }
1347   } {
1348     \peek_charcode_remove:NTF ? {
1349       \__stex_modules_invoke_symbol:nn { #1 }
1350     } {
1351       \msg_error:nnx{stex}{error/syntax}{
1352         ?~or~!~expected~after~
1353         \c_backslash_str STEXModule{#1}
1354       }
1355     }
1356   }
1357 }
1358
1359 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1360   \str_set:Nn #2 { #1 }
1361 }
1362
1363 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1364   \stex_invoke_symbol:n{#1?#2}
1365 }
```

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

`\stex_activate_module:n`

```
1366 \bool_new:N \l_stex_in_meta_bool
1367 \bool_set_false:N \l_stex_in_meta_bool
1368 \cs_new_protected:Nn \stex_activate_module:n {
1369   \stex_debug:nn{modules}{Activating~module~#1}
1370   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1371     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1372   }
1373   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1374     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1375     \use:c{ c_stex_module_#1_code }
1376   }
1377 }
```

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

```
1378 </package>
```

Chapter 29

sTeX -Module Inheritance Implementation

```
1379 <*package>
1380
1381 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1382
```

29.1 SMS Mode

```
1383 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1384 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1385 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1386 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1387
1388 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1389   \makeatletter
1390   \makeatother
1391   \ExplSyntaxOn
1392   \ExplSyntaxOff
1393 }
1394
1395 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1396   \symdef
1397   \importmodule
1398   \notation
1399   \symdecl
1400   \STEXexport
1401 }
1402
1403 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1404   \tl_to_str:n {
1405     module,
1406     @module
```



```
1407 }
1408 }
```

(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
1409 \bool_new:N \g__stex_smsmode_bool
1410 \bool_set_false:N \g__stex_smsmode_bool
1411 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1412   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1413 }
```

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

```
\_stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\_stex_smsmode_if_catcodes:TF
1414 \bool_new:N \g__stex_smsmode_catcode_bool
1415 \bool_set_false:N \g__stex_smsmode_catcode_bool
1416 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1417   \bool_if:NTF \g__stex_smsmode_catcode_bool
1418   \prg_return_true: \prg_return_false:
1419 }
```

(End definition for `_stex_smsmode_if_catcodes:TF`.)

```
\stex_smsmode_set_codes:
1420 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1421   \stex_if_smsmode:T {
1422     \_stex_smsmode_if_catcodes:F {
1423       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1424       \exp_after:wN \char_gset_active_eq:NN
1425       \c_backslash_str \_stex_smsmode_cs:
1426       \tex_global:D \char_set_catcode_active:N \
1427       \tex_global:D \char_set_catcode_other:N $
1428       \tex_global:D \char_set_catcode_other:N ^
1429       \tex_global:D \char_set_catcode_other:N _
1430       \tex_global:D \char_set_catcode_other:N &
1431       \tex_global:D \char_set_catcode_other:N ##
1432     }
1433   }
1434 } \iffalse $ \fi % to make syntax highlighting work again
```

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

```
\_stex_smsmode_unset_codes: Sets category code scheme back from the one used in SMS mode.
1435 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1436   \_stex_smsmode_if_catcodes:T {
1437     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1438     \exp_after:wN \tex_global:D \exp_after:wN
1439     \char_set_catcode_escape:N \c_backslash_str
1440     \tex_global:D \char_set_catcode_math_toggle:N $
1441     \tex_global:D \char_set_catcode_math_superscript:N ^
1442     \tex_global:D \char_set_catcode_math_subscript:N _
1443     \tex_global:D \char_set_catcode_alignment:N &
1444     \tex_global:D \char_set_catcode_parameter:N ##
1445   }
1446 } \iffalse $ \fi % to make syntax highlighting work again
```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1447 \cs_new_protected:Nn \stex_in_smsmode:nn {
1448   \vbox_set:Nn \l_tmpa_box {
1449     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1450     \bool_gset_true:N \g__stex_smsmode_bool
1451     \stex_smsmode_set_codes:
1452     #2
1453     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1454     \stex_if_smsmode:F {
1455       \__stex_smsmode_unset_codes:
1456     }
1457   }
1458   \box_clear:N \l_tmpa_box
1459 }

```

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

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

```

1460 \cs_new_protected:Nn \_stex_smsmode_cs: {
1461   \str_clear:N \l_tmpa_str
1462   \peek_analysis_map_inline:n {
1463     % #1: token (one expansion)
1464     % #2: charcode
1465     % #3 catcode
1466     \token_if_eq_charcode:NNTF ##3 B {
1467       % token is a letter
1468       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1469     } {
1470       \str_if_empty:NTF \l_tmpa_str {
1471         % we don't allow (or need) single non-letter CSs
1472         % for now
1473         \peek_analysis_map_break:
1474       }{
1475         \str_if_eq:onTF \l_tmpa_str { begin } {
1476           \peek_analysis_map_break:n {
1477             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1478           }
1479         } {
1480           \str_if_eq:onTF \l_tmpa_str { end } {
1481             \peek_analysis_map_break:n {
1482               \exp_after:wN \_stex_smsmode_checkend:n ##1
1483             }
1484           } {
1485             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1486             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1487               \g_stex_smsmode_allowedmacros_tl
1488               { \use:c{\l_tmpa_str} } {
1489               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1490               \peek_analysis_map_break:n {
1491                 \exp_after:wN \l_tmpa_tl ##1
1492               }
1493             }
1494           }
1495         }
1496       }
1497     }
1498   }
1499 }

```

```

1493     } {
1494         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1495         \g_stex_smsmode_allowedmacros_escape_tl
1496         { \use:c{\l_tmpa_str} } {
1497             \__stex_smsmode_unset_codes:
1498             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1499             % TODO \__stex_smsmode_rescan_cs:
1500             \int_compare:nNnTF {##2} = {92} {
1501                 \peek_analysis_map_break:n {
1502                     \__stex_smsmode_unset_codes:
1503                     \__stex_smsmode_rescan_cs:
1504                 }
1505             } {
1506                 \peek_analysis_map_break:n {
1507                     \exp_after:wN \l_tmpa_tl ##1
1508                 }
1509             }
1510         } {
1511             \int_compare:nNnTF {##2} = {92} {
1512                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1513             } {
1514                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1515             }
1516         }
1517     }
1518 }
1519 }
1520 }
1521 }
1522 }
1523 }

```

(End definition for __stex_smsmode_cs:.)

__stex_smsmode_rescan_cs: If the last token gobbled by \stex_smsmode_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```

1524 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1525     \str_clear:N \l_tmpb_str
1526     \peek_analysis_map_inline:n {
1527         \token_if_eq_charcode:NNTF ##3 B {
1528             % token is a letter
1529             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1530         } {
1531             \peek_analysis_map_break:n {
1532                 \exp_after:wN \use:c \exp_after:wN {
1533                     \exp_after:wN \l_tmpa_str\exp_after:wN
1534                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1535             }
1536         }
1537     }
1538 }

```

(End definition for __stex_smsmode_rescan_cs:.)

`__stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1539 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1540   \str_set:Nn \l_tmpa_str { #1 }
1541   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1542     \__stex_smsmode_unset_codes:
1543     \begin{#1}
1544   }
1545 }
```

(End definition for `__stex_smsmode_checkbegin:n`.)

`__stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1552 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1553   \str_set:Nn \l_tmpa_str { #1 }
1554   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1555     \end{#1}
1556   }
1557 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1552 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1553 \cs_new_protected:Nn \stex_import_module_uri:nn {
1554   \str_set:Nx \l_stex_import_archive_str { #1 }
1555   \str_set:Nn \l_stex_import_path_str { #2 }
1556
1557   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1558   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1559   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1560
1561   \stex_modules_current_namespace:
1562   \bool_lazy_all:nTF {
1563     {\str_if_empty_p:N \l_stex_import_archive_str}
1564     {\str_if_empty_p:N \l_stex_import_path_str}
1565     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1566   }{
1567     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1568     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1569   }{
1570     \str_if_empty:NT \l_stex_import_archive_str {
1571       \prop_if_exist:NT \l_stex_current_repository_prop {
1572         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1573       }
1574     }
1575     \str_if_empty:NTF \l_stex_import_archive_str {
1576       \str_if_empty:NF \l_stex_import_path_str {
1577         \str_set:Nx \l_stex_import_ns_str {
1578           \l_stex_module_ns_str / \l_stex_import_path_str
1579         }
1580       }
1581     }
```

```

1581     }{
1582       \stex_require_repository:n \l_stex_import_archive_str
1583       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
1584       \l_stex_import_ns_str
1585       \str_if_empty:NF \l_stex_import_path_str {
1586         \str_set:Nx \l_stex_import_ns_str {
1587           \l_stex_import_ns_str / \l_stex_import_path_str
1588         }
1589       }
1590     }
1591   }
1592 }

```

(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 1593 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1594 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1595 \str_new:N \l_stex_import_path_str
                           1596 \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:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1597 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1598   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1599
1600     % archive
1601     \str_set:Nx \l_tmpa_str { #2 }
1602     \str_if_empty:NTF \l_tmpa_str {
1603       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1604     } {
1605       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1606       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1607       \seq_put_right:Nn \l_tmpa_seq { source }
1608     }
1609
1610     % path
1611     \str_set:Nx \l_tmpb_str { #3 }
1612     \str_if_empty:NTF \l_tmpb_str {
1613       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1614
1615       \ltx@ifpackageloaded{babel} {
1616         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1617           { \language } \l_tmpb_str {
1618           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1619         }
1620       } {
1621         \str_clear:N \l_tmpb_str
1622       }
1623
1624       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1625       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1626         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1627 }{
1628   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1629   \IfFileExists{ \l_tmpa_str.tex }{
1630     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1631   }{
1632     % try english as default
1633     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1634     \IfFileExists{ \l_tmpa_str.en.tex }{
1635       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1636     }{
1637       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1638     }
1639   }
1640 }
1641
1642 } {
1643   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1644   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1645
1646   \ltx@ifpackageloaded{babel} {
1647     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1648       { \language } \l_tmpb_str {
1649       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1650     }
1651   } {
1652     \str_clear:N \l_tmpb_str
1653   }
1654
1655   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1656
1657   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1658   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1659     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1660   }{
1661     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1662     \IfFileExists{ \l_tmpa_str/#4.tex }{
1663       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1664     }{
1665       % try english as default
1666       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1667       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1668         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1669       }{
1670         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1671         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1672           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1673         }{
1674           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1675           \IfFileExists{ \l_tmpa_str.tex }{
1676             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1677           }{
1678             % try english as default
1679             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1680             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1681         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1682     }{
1683         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1684     }
1685 }
1686 }
1687 }
1688 }
1689 }
1690 }
1691
1692 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1693     \seq_clear:N \l_stex_all_modules_seq
1694     \str_clear:N \l_stex_current_module_str
1695     \str_set:Nx \l_tmpb_str { #2 }
1696     \str_if_empty:NF \l_tmpb_str {
1697         \stex_set_current_repository:n { #2 }
1698     }
1699     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1700     \input { \g__stex_importmodule_file_str }
1701 }
1702
1703 \stex_if_module_exists:nF { #1 ? #4 } {
1704     \msg_error:nnx{stex}{error/unknownmodule}{
1705         #1?#4~(in~file~\g__stex_importmodule_file_str)
1706     }
1707 }
1708 }
1709 \stex_activate_module:n { #1 ? #4 }
1710 }

```

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

`\importmodule`

```

1711 \NewDocumentCommand \importmodule { 0{} m } {
1712     \stex_import_module_uri:nn { #1 } { #2 }
1713     \stex_debug:nn{modules}{Importing~module:~
1714         \l_stex_import_ns_str ? \l_stex_import_name_str
1715     }
1716     \stex_if_smsmode:F {
1717         \stex_import_require_module:nnnn
1718         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1719         { \l_stex_import_path_str } { \l_stex_import_name_str }
1720         \stex_annotate_invisible:nnn
1721         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1722     }
1723     \exp_args:Nx \stex_add_to_current_module:n {
1724         \stex_import_require_module:nnnn
1725         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1726         { \l_stex_import_path_str } { \l_stex_import_name_str }
1727     }
1728     \exp_args:Nx \stex_add_import_to_current_module:n {
1729         \l_stex_import_ns_str ? \l_stex_import_name_str
1730     }

```

```

1731 \stex_smsmode_set_codes:
1732 }
1733 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1734 \NewDocumentCommand \usemodule { 0{} m } {
1735   \stex_if_smsmode:F {
1736     \stex_import_module_uri:nn { #1 } { #2 }
1737     \stex_import_require_module:nnnn
1738     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1739     { \l_stex_import_path_str } { \l_stex_import_name_str }
1740     \stex_annotate_invisible:nnn
1741     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1742   }
1743   \stex_smsmode_set_codes:
1744 }

```

(End definition for \usemodule. This function is documented on page 33.)

```

1745 \endpackage

```


Chapter 30

STEX -Symbols Implementation

```
1746 <*package>
1747
1748 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1749
```

Warnings and error messages

```
1750
```

30.1 Symbol Declarations

```
1751 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1752 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1753 \NewDocumentCommand \STEXsymbol { m } {
1754   \stex_get_symbol:n { #1 }
1755   \exp_args:No
1756   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1757 }
```

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

symdecl arguments:

```
1758 \keys_define:nn { stex / symdecl } {
1759   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1760   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1761   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1762   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1763   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1764   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1765   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1766   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1767 }
```

```

1768
1769 \bool_new:N \l_stex_symdecl_make_macro_bool
1770
1771 \cs_new_protected:Nn \__stex_symdecl_args:n {
1772   \str_clear:N \l_stex_symdecl_name_str
1773   \str_clear:N \l_stex_symdecl_args_str
1774   \bool_set_false:N \l_stex_symdecl_local_bool
1775   \tl_clear:N \l_stex_symdecl_type_tl
1776   \tl_clear:N \l_stex_symdecl_definiens_tl
1777
1778   \keys_set:nn { stex / symdecl } { #1 }
1779 }

```

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

```

1780
1781 \NewDocumentCommand \symdecl { s O{} m } {
1782   \__stex_symdecl_args:n { #2 }
1783   \IfBooleanTF #1 {
1784     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1785   } {
1786     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1787   }
1788   \stex_symdecl_do:n { #3 }
1789   \stex_smsmode_set_codes:
1790 }
1791 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1792 \cs_new_protected:Nn \stex_symdecl_do:n {
1793   \stex_if_in_module:F {
1794     % TODO throw error? some default namespace?
1795   }
1796
1797   \str_if_empty:NT \l_stex_symdecl_name_str {
1798     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1799   }
1800
1801   \prop_if_exist:cT { l_stex_symdecl_
1802     \l_stex_current_module_str ?
1803     \l_stex_symdecl_name_str
1804     _prop
1805   }{
1806     % TODO throw error (beware of circular dependencies)
1807   }
1808
1809   \prop_clear:N \l_tmpa_prop
1810   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1811   \seq_clear:N \l_tmpa_seq
1812   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1813   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1814

```

```

1815 \exp_args:No \stex_add_constant_to_current_module:n {
1816   \l_stex_symdecl_name_str
1817 }
1818
1819 % arity/args
1820 \int_zero:N \l_tmpb_int
1821
1822 \bool_set_true:N \l_tmpa_bool
1823 \str_map_inline:Nn \l_stex_symdecl_args_str {
1824   \token_case_meaning:NnF ##1 {
1825     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1826     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1827     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1828     {\tl_to_str:n a} {
1829       \bool_set_false:N \l_tmpa_bool
1830       \int_incr:N \l_tmpb_int
1831     }
1832     {\tl_to_str:n B} {
1833       \bool_set_false:N \l_tmpa_bool
1834       \int_incr:N \l_tmpb_int
1835     }
1836   }{
1837     \msg_set:nnn{stex}{error/wrongargs}{
1838       args~value~in~symbol~declaration~for~
1839       \l_stex_current_module_str ?
1840       \l_stex_symdecl_name_str ~
1841       needs~to~be~
1842       i,~a,~b~or~B,~but~##1~given
1843     }
1844     \msg_error:nn{stex}{error/wrongargs}
1845   }
1846 }
1847 \bool_if:NTF \l_tmpa_bool {
1848   % possibly numeric
1849   \str_if_empty:NTF \l_stex_symdecl_args_str {
1850     \prop_put:Nnn \l_tmpa_prop { args } {}
1851     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1852   }{
1853     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1854     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1855     \str_clear:N \l_tmpa_str
1856     \int_step_inline:nn \l_tmpa_int {
1857       \str_put_right:Nn \l_tmpa_str i
1858     }
1859     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1860   }
1861 } {
1862   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1863   \prop_put:Nnx \l_tmpa_prop { arity }
1864     { \str_count:N \l_stex_symdecl_args_str }
1865 }
1866 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1867
1868

```

```

1869 % semantic macro
1870
1871 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1872   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1873     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1874   } }
1875
1876   \bool_if:NF \l_stex_symdecl_local_bool {
1877     \exp_args:Nx \stex_add_to_current_module:n {
1878       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1879         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1880       } }
1881     }
1882   }
1883 }
1884
1885 % add to all symbols
1886
1887 \bool_if:NF \l_stex_symdecl_local_bool {
1888   \exp_args:Nx \stex_add_to_current_module:n {
1889     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1890       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1891     }
1892   }
1893 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1894 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1895 %   }
1896 }
1897
1898 \stex_debug:nn{symbols}{New~symbol:~
1899   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1900   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1901   Args:~\prop_item:Nn \l_tmpa_prop { args }
1902 }
1903
1904 % circular dependencies require this:
1905
1906 \prop_if_exist:cF {
1907   l_stex_symdecl_
1908   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1909   _prop
1910 } {
1911   \prop_set_eq:cN {
1912     l_stex_symdecl_
1913     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1914     _prop
1915   } \l_tmpa_prop
1916 }
1917
1918 \seq_clear:c {
1919   l_stex_symdecl_
1920   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1921   _notations
1922 }

```

```

1923
1924 \bool_if:NF \l_stex_symdecl_local_bool {
1925   \exp_args:Nx
1926   \stex_add_to_current_module:n {
1927     \seq_clear:c {
1928       l_stex_symdecl_
1929       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1930       _notations
1931     }
1932     \prop_set_from_keyval:cn {
1933       l_stex_symdecl_
1934       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1935       _prop
1936     } {
1937       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1938       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1939       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1940       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1941       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1942       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1943     }
1944   }
1945 }
1946
1947 \stex_if_smsmode:TF {
1948   \bool_if:NF \l_stex_symdecl_local_bool {
1949     % \exp_args:Nx \stex_add_to_sms:n {
1950     %   \prop_set_from_keyval:cn {
1951     %     l_stex_symdecl_
1952     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1953     %     _prop
1954     %   } {
1955     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1956     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1957     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1958     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1959     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1960     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1961     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1962     %   }
1963     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1964     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1965     %   }
1966     % }
1967   }
1968 }{
1969   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1970     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971   }
1972   \stex_if_do_html:T {
1973     \stex_annotate_invisible:nnn {symdecl} {
1974       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1975     } {
1976       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st

```

```

1977     \stex_annotate_invisible:nnn{args}{}{
1978     \prop_item:Nn \l_tmpa_prop { args }
1979   }
1980   \stex_annotate_invisible:nnn{macroname}{#1}{}
1981   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1982     \stex_annotate_invisible:nnn{definiens}{}
1983     { $\l_stex_symdecl_definiens_tl$ }
1984   }
1985 }
1986 }
1987 }
1988 }

```

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

`\stex_get_symbol:n`

```

1989 \str_new:N \l_stex_get_symbol_uri_str
1990
1991 \cs_new_protected:Nn \stex_get_symbol:n {
1992   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1993     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1994   }{
1995     % argument is a string
1996     % is it a command name?
1997     \cs_if_exist:cTF { #1 }{
1998       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1999       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2000       \str_if_empty:NNTF \l_tmpa_str {
2001         \exp_args:Nx \cs_if_eq:NNTF {
2002           \tl_head:N \l_tmpa_tl
2003         } \stex_invoke_symbol:n {
2004           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2005         }{
2006           \__stex_symdecl_get_symbol_from_string:n { #1 }
2007         }
2008       } {
2009         \__stex_symdecl_get_symbol_from_string:n { #1 }
2010       }
2011     }{
2012       % argument is not a command name
2013       \__stex_symdecl_get_symbol_from_string:n { #1 }
2014       % \l_stex_all_symbols_seq
2015     }
2016   }
2017 }
2018
2019 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2020   \str_set:Nn \l_tmpa_str { #1 }
2021   \bool_set_false:N \l_tmpa_bool
2022   \stex_if_in_module:T {
2023     \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2024       \bool_set_true:N \l_tmpa_bool
2025       \str_set:Nx \l_stex_get_symbol_uri_str {
2026         \l_stex_current_module_str ? #1

```

```

2027     }
2028   }
2029 }
2030 \bool_if:NF \l_tmpa_bool {
2031   \tl_set:Nn \l_tmpa_tl {
2032     \msg_set:nnn{stex}{error/unknownsymbol}{
2033       No~symbol~#1~found!
2034     }
2035     \msg_error:nn{stex}{error/unknownsymbol}
2036   }
2037   \str_set:Nn \l_tmpa_str { #1 }
2038   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2039   \seq_map_inline:Nn \l_stex_all_symbols_seq {
2040     \str_set:Nn \l_tmpb_str { ##1 }
2041     \str_if_eq:eeT { \l_tmpa_str } {
2042       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2043     } {
2044       \seq_map_break:n {
2045         \tl_set:Nn \l_tmpa_tl {
2046           \str_set:Nn \l_stex_get_symbol_uri_str {
2047             ##1
2048           }
2049         }
2050       }
2051     }
2052   }
2053   \l_tmpa_tl
2054 }
2055 }
2056
2057 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2058   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2059   { \tl_tail:N \l_tmpa_tl }
2060   \tl_if_single:NTF \l_tmpa_tl {
2061     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2062       \exp_after:wN \str_set:Nn \exp_after:wN
2063       \l_stex_get_symbol_uri_str \l_tmpa_tl
2064     }{
2065       % TODO
2066       % tail is not a single group
2067     }
2068   }{
2069     % TODO
2070     % tail is not a single group
2071   }
2072 }

```

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

30.2 Notations

```

2073 <@@=stex_notation>

```

notation arguments:

```

2074 \keys_define:nn { stex / notation } {
2075   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2076   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2077   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2078   op        .tl_set:N   = \l__stex_notation_op_tl ,
2079   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2080   primary   .default:n  = {true} ,
2081   unknown   .code:n     = \str_set:Nx
2082             \l__stex_notation_variant_str \l_keys_key_str
2083 }
2084
2085 \cs_new_protected:Nn \stex_notation_args:n {
2086   \str_clear:N \l__stex_notation_lang_str
2087   \str_clear:N \l__stex_notation_variant_str
2088   \str_clear:N \l__stex_notation_prec_str
2089   \tl_clear:N \l__stex_notation_op_tl
2090   \bool_set_false:N \l__stex_notation_primary_bool
2091
2092   \keys_set:nn { stex / notation } { #1 }
2093 }

```

\notation

```

2094 \NewDocumentCommand \notation { 0{ } m } {
2095   \stex_notation_args:n { #1 }
2096   \tl_clear:N \l_stex_symdecl_definiens_tl
2097   \stex_get_symbol:n { #2 }
2098   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2099 }
2100 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

2101 \cs_new_protected:Nn \stex_notation_do:nn {
2102   \let\l_stex_current_symbol_str\relax
2103   \prop_set_eq:Nc \l_tmpa_prop {
2104     l_stex_symdecl_ #1 _prop
2105   }
2106
2107   \prop_clear:N \l_tmpb_prop
2108   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2109   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2110   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2111
2112   % precedences
2113   \seq_clear:N \l_tmpb_seq
2114   \exp_args:NNno
2115   \str_if_empty:NTF \l__stex_notation_prec_str {
2116     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2117     \int_compare:nNnTF \l_tmpa_str = 0 {
2118       \exp_args:NNnx
2119       \prop_put:Nno \l_tmpb_prop { opprec }
2120       { \neginfprec }
2121     }{
2122       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }

```



```

2123     }
2124   } {
2125     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2126       \exp_args:NNnx
2127       \prop_put:Nno \l_tmpb_prop { opprec }
2128       { \neginfprec }
2129       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2130       \int_step_inline:nn { \l_tmpa_str } {
2131         \exp_args:NNx
2132         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2133       }
2134     }{
2135       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2136       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2137         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2138         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2139           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2140             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2141           \seq_map_inline:Nn \l_tmpa_seq {
2142             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2143           }
2144         }
2145         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2146       }{
2147         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2148         \int_compare:nNnTF \l_tmpa_str = 0 {
2149           \exp_args:NNnx
2150           \prop_put:Nno \l_tmpb_prop { opprec }
2151           { \infprec }
2152         }{
2153           \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2154         }
2155       }
2156     }
2157   }
2158
2159   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2160   \int_step_inline:nn { \l_tmpa_str } {
2161     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2162       \exp_args:NNx
2163       \seq_put_right:Nn \l_tmpb_seq {
2164         \prop_item:Nn \l_tmpb_prop { opprec }
2165       }
2166     }
2167   }
2168
2169   \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2170   \tl_clear:N \l_tmpa_tl
2171
2172   \int_compare:nNnTF \l_tmpa_str = 0 {
2173     \exp_args:NNe
2174     \cs_set:Npn \l__stex_notation_macrocode_cs {
2175       \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2176       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2177         { \prop_item:Nn \l_tmpb_prop { opprec } }
2178         { \exp_not:n { #2 } }
2179     }
2180     \__stex_notation_final:
2181 }{
2182     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2183     \str_if_in:NnTF \l_tmpb_str b {
2184         \exp_args:Nne \use:nn
2185         {
2186             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2187             \cs_set:Npn \l_tmpa_str } { {
2188                 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2189                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2190                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2191                 { \exp_not:n { #2 } }
2192             } }
2193         }{
2194             \str_if_in:NnTF \l_tmpb_str B {
2195                 \exp_args:Nne \use:nn
2196                 {
2197                     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2198                     \cs_set:Npn \l_tmpa_str } { {
2199                         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2200                         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2201                         { \prop_item:Nn \l_tmpb_prop { opprec } }
2202                         { \exp_not:n { #2 } }
2203                     } }
2204                 }{
2205                     \exp_args:Nne \use:nn
2206                     {
2207                         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2208                         \cs_set:Npn \l_tmpa_str } { {
2209                             \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2210                             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2211                             { \prop_item:Nn \l_tmpb_prop { opprec } }
2212                             { \exp_not:n { #2 } }
2213                         } }
2214                     }
2215                 }
2216
2217         \int_zero:N \l_tmpa_int
2218         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2219         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2220         \__stex_notation_arguments:
2221     }
2222 }

```

(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

```

2223 \cs_new_protected:Nn \__stex_notation_arguments: {
2224     \int_incr:N \l_tmpa_int
2225     \str_if_empty:NnTF \l_tmpa_str {
2226         \__stex_notation_final:

```

```

2227 }{
2228   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2229   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2230   \str_if_eq:VnTF \l_tmpb_str a {
2231     \__stex_notation_argument_assoc:n
2232   }{
2233     \str_if_eq:VnTF \l_tmpb_str B {
2234       \__stex_notation_argument_assoc:n
2235     }{
2236       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2237       \tl_put_right:Nx \l_tmpa_tl {
2238         { \stex_term_math_arg:nnn
2239           { \int_use:N \l_tmpa_int }
2240           { \l_tmpb_str }
2241           { ####\int_use:N \l_tmpa_int }
2242         }
2243       }
2244       \__stex_notation_arguments:
2245     }
2246   }
2247 }
2248 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2249 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2250   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2251   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2252   \tl_put_right:Nx \l_tmpa_tl {
2253     { \stex_term_math_assoc_arg:nnnn
2254       { \int_use:N \l_tmpa_int }
2255       { \l_tmpb_str }
2256       \exp_args:No \exp_not:n
2257       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2258       { ####\int_use:N \l_tmpa_int }
2259     }
2260   }
2261   \__stex_notation_arguments:
2262 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2263 \cs_new_protected:Nn \__stex_notation_final: {
2264   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2265   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2266   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2267   \exp_args:Nne \use:nn
2268   {
2269     \cs_generate_from_arg_count:cNnn {
2270       stex_notation_ \l_tmpa_str \c_hash_str
2271       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2272       _cs

```

```

2273     }
2274     \cs_set:Npn \l_tmpb_str } { {
2275         \exp_after:wN \exp_after:wN \exp_after:wN
2276         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2277         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2278     } }
2279
2280 \tl_if_empty:NF \l__stex_notation_op_tl {
2281     \cs_set:cpx {
2282         stex_op_notation_ \l_tmpa_str \c_hash_str
2283         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2284         _cs
2285     } {
2286         \stex_term_oms:nnn {
2287             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2288             \l__stex_notation_lang_str
2289         }{
2290             \l_tmpa_str
2291         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2292     }
2293 }
2294
2295 \exp_args:Ne
2296 \stex_add_to_current_module:n {
2297     \cs_generate_from_arg_count:cNnn {
2298         stex_notation_ \l_tmpa_str \c_hash_str
2299         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2300         _cs
2301     } \cs_set:Npn {\l_tmpb_str} {
2302         \exp_after:wN \exp_after:wN \exp_after:wN
2303         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2304         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2305     }
2306 \tl_if_empty:NF \l__stex_notation_op_tl {
2307     \cs_set:cpn {
2308         stex_op_notation_ \l_tmpa_str \c_hash_str
2309         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2310         _cs
2311     } {
2312         \stex_term_oms:nnn {
2313             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2314             \l__stex_notation_lang_str
2315         }{
2316             \l_tmpa_str
2317         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2318     }
2319 }
2320 }
2321
2322 \seq_put_right:cx {
2323     l_stex_symdecl_
2324     \prop_item:Nn \l_tmpb_prop { symbol }
2325     _notations
2326 } {

```

```

2327 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2328 }
2329
2330 \stex_debug:nn{symbols}{
2331   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2332   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2333   Operator~precedence:~
2334   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2335   Argument~precedences:~
2336   \seq_use:Nn \l_tmpa_seq {,~}^^J
2337   Notation: \cs_meaning:c {
2338     stex_notation_ \l_tmpa_str \c_hash_str
2339     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2340     _cs
2341   }
2342 }
2343
2344 \prop_set_eq:cN {
2345   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2346   \c_hash_str \l__stex_notation_lang_str _prop
2347 } \l_tmpb_prop
2348
2349 \exp_args:Ne
2350 \stex_add_to_current_module:n {
2351   \seq_put_right:cn {
2352     l_stex_symdecl_
2353     \prop_item:Nn \l_tmpb_prop { symbol }
2354     _notations
2355   } {
2356     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2357   }
2358   \prop_set_from_keyval:cn {
2359     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2360     \c_hash_str \l__stex_notation_lang_str _prop
2361   } {
2362     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2363     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2364     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2365     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2366     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2367   }
2368 }
2369
2370 \stex_if_smsmode:TF {
2371   \stex_smsmode_set_codes:
2372   % \exp_args:Nx \stex_add_to_sms:n {
2373   %   \prop_set_from_keyval:cn {
2374   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2375   %     \c_hash_str \l__stex_notation_lang_str _prop
2376   %   } {
2377   %     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2378   %     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2379   %     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2380   %     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,

```

```

2381 %      argprec = \prop_item:Nn \l_tmpb_prop { argprec } ,
2382 %    }
2383 %  }
2384 }{
2385
2386 % HTML annotations
2387 \stex_if_do_html:T {
2388   \stex_annotate_invisible:nnn { notation }
2389   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2390     \stex_annotate_invisible:nnn { notationfragment }
2391     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2392     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2393     \stex_annotate_invisible:nnn { precedence }
2394     { \prop_item:Nn \l_tmpb_prop { opprec };
2395       \seq_use:Nn \l_tmpa_seq { x }
2396     }{}
2397
2398     \int_zero:N \l_tmpa_int
2399     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2400     \tl_clear:N \l_tmpa_tl
2401     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2402       \int_incr:N \l_tmpa_int
2403       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2404       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2405       \str_if_eq:VnTF \l_tmpb_str a {
2406         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2407           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2408           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2409         } }
2410       }{
2411         \str_if_eq:VnTF \l_tmpb_str B {
2412           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2413             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2414             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2415           } }
2416         }{
2417           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2418             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2419           } }
2420         }
2421       }
2422     }
2423     \stex_annotate_invisible:nnn { notationcomp }{}{
2424       \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2425       $ \exp_args:Nno \use:nn { \use:c {
2426         stex_notation_ \l_stex_current_symbol_str
2427         \c_hash_str \l__stex_notation_variant_str
2428         \c_hash_str \l__stex_notation_lang_str _cs
2429       } } { \l_tmpa_tl } $
2430     }
2431   }
2432 }
2433 }
2434 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2435 \keys_define:nn { stex / setnotation } {
2436   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2437   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2438   unknown   .code:n      = \str_set:Nx
2439             \l__stex_notation_variant_str \l_keys_key_str
2440 }
2441
2442 \cs_new_protected:Nn \_stex_setnotation_args:n {
2443   \str_clear:N \l__stex_notation_lang_str
2444   \str_clear:N \l__stex_notation_variant_str
2445   \keys_set:nn { stex / setnotation } { #1 }
2446 }
2447
2448 \NewDocumentCommand \setnotation {m m} {
2449   \stex_get_symbol:n { #1 }
2450   \_stex_setnotation_args:n { #2 }
2451   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2452     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2453     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2454       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2455     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2456       { \c_hash_str }
2457     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2458       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2459     \exp_args:Nx \stex_add_to_current_module:n {
2460       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2461         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2462       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2463         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2464       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2465         { \c_hash_str }
2466     }
2467     \stex_debug:nn {notations}{
2468       Setting~default~notation~
2469       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2470       \l_stex_get_symbol_uri_str \
2471       \expandafter\meaning\csname
2472       l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2473     }
2474   }{
2475     % todo throw error
2476   }
2477 }
2478

```

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

\symdef

```

2479 \keys_define:nn { stex / symdef } {
2480   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2481   local     .bool_set:N = \l_stex_symdecl_local_bool ,

```

```

2482 args      .str_set_x:N = \l_stex_symdecl_args_str ,
2483 type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2484 def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2485 op        .tl_set:N    = \l__stex_notation_op_tl ,
2486 lang      .str_set_x:N = \l__stex_notation_lang_str ,
2487 variant   .str_set_x:N = \l__stex_notation_variant_str ,
2488 prec      .str_set_x:N = \l__stex_notation_prec_str ,
2489 unknown   .code:n      = \str_set:Nx
2490           \l__stex_notation_variant_str \l_keys_key_str
2491 }
2492
2493 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2494   \str_clear:N \l_stex_symdecl_name_str
2495   \str_clear:N \l_stex_symdecl_args_str
2496   \bool_set_false:N \l_stex_symdecl_local_bool
2497   \tl_clear:N \l_stex_symdecl_type_tl
2498   \tl_clear:N \l_stex_symdecl_definiens_tl
2499   \str_clear:N \l__stex_notation_lang_str
2500   \str_clear:N \l__stex_notation_variant_str
2501   \str_clear:N \l__stex_notation_prec_str
2502   \tl_clear:N \l__stex_notation_op_tl
2503
2504   \keys_set:nn { stex / symdef } { #1 }
2505 }
2506
2507 \NewDocumentCommand \symdef { 0{} m } {
2508   \__stex_notation_symdef_args:n { #1 }
2509   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2510   \stex_symdecl_do:n { #2 }
2511   \exp_args:Nx \stex_notation_do:nn {
2512     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2513   }
2514 }
2515 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2516 </package>

```


Chapter 31

STEX -Terms Implementation

```
2517 <*package>
2518
2519 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2520
2521 <@@=stex_terms>
2522
2523   Warnings and error messages
2524   \msg_new:nnn{stex}{error/nonotation}{
2525     Symbol~#1~invoked,~but~has~no~notation~#2!
2526   }
2527   \msg_new:nnn{stex}{error/notationarg}{
2528     Error~in~parsing~notation~#1
2529   }
2530   \msg_new:nnn{stex}{error/noop}{
2531     Symbol~#1~has~no~operator~notation~for~notation~#2
2532   }
2533
```

31.1 Symbol Invocations

Arguments:

```
2532 \keys_define:nn { stex / terms } {
2533   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2534   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2535   unknown .code:n = \str_set:Nx
2536     \l__stex_terms_variant_str \l_keys_key_str
2537 }
2538
2539 \cs_new_protected:Nn \__stex_terms_args:n {
2540   \str_clear:N \l__stex_terms_lang_str
2541   \str_clear:N \l__stex_terms_variant_str
2542   \str_clear:N \l__stex_terms_prec_str
2543   \tl_clear:N \l__stex_terms_op_tl
2544
2545   \keys_set:nn { stex / terms } { #1 }
```

2546 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2547 \cs_new_protected:Nn \stex_invoke_symbol:n {
2548   \if_mode_math:
2549     \exp_after:wN \__stex_terms_invoke_math:n
2550   \else:
2551     \exp_after:wN \__stex_terms_invoke_text:n
2552   \fi: { #1 }
2553 }
```

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

__stex_terms_invoke_math:n

```
2554 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2555   \peek_charcode_remove:NTF ! {
2556     \peek_charcode:NTF [ {
2557       \__stex_terms_invoke_op:nw { #1 }
2558     }{
2559       \peek_charcode_remove:NTF ! {
2560         \peek_charcode:NTF [ {
2561           \__stex_terms_invoke_op_custom:nw
2562         }{
2563           % TODO throw error
2564         }
2565       }{
2566         \__stex_terms_invoke_op:nw { #1 } []
2567       }
2568     }{
2569       \peek_charcode_remove:NTF * {
2570         \__stex_terms_invoke_text:n { #1 }
2571       }{
2572         \peek_charcode:NTF [ {
2573           \__stex_terms_invoke_math:nw { #1 }
2574         }{
2575           \__stex_terms_invoke_math:nw { #1 } []
2576         }
2577       }
2578     }
2579   }
2580 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2581 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2582   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2583     \stex_highlight_term:nn{#1}{#2}
2584   }
2585 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2586 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2587   \_stex_terms_args:n { #2 }
2588   \cs_if_exist:cTF {
2589     stex_op_notation_ #1 \c_hash_str
2590     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2591   }{
2592     \csname stex_op_notation_ #1 \c_hash_str
2593       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2594     \endcsname
2595   }{
2596     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2597   }
2598 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2599 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2600   \_stex_terms_args:n { #2 }
2601   \seq_if_empty:cTF {
2602     l_stex_symdecl_ #1 _notations
2603   } {
2604     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2605   } {
2606     \seq_if_in:cxTF {
2607       l_stex_symdecl_ #1 _notations
2608     }
2609     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2610       \str_set:Nn \l_stex_current_symbol_str { #1 }
2611       \use:c{
2612         stex_notation_ #1 \c_hash_str
2613         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2614         _cs
2615       }
2616     }{
2617       \str_if_empty:NTF \l__stex_terms_variant_str {
2618         \str_if_empty:NTF \l__stex_terms_lang_str {
2619           \seq_get_left:cN {
2620             l_stex_symdecl_ #1 _notations
2621           } \l_tmpa_str
2622           \str_set:Nn \l_stex_current_symbol_str { #1 }
2623           \use:c{
2624             stex_notation_ #1 \c_hash_str \l_tmpa_str
2625             _cs
2626           }
2627         }{
2628           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2629             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2630           }
2631         }
2632       }{
2633         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2634           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2635     }
2636   }
2637 }
2638 }
2639 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2640 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2641   \peek_charcode_remove:NTF ! {
2642     \stex_term_custom:nn { #1 } { }
2643   }{
2644     \prop_set_eq:Nc \l_tmpa_prop {
2645       l_stex_symdecl_ #1 _prop
2646     }
2647     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2648     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2649   }
2650 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2651 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2652 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2653 \int_new:N \l__stex_terms_downprec
2654 \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
2655 \tl_set:Nn \l__stex_terms_left_bracket_str (
2656 \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

```

2657 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2658   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2659     \bool_set_false:N \l__stex_terms_brackets_done_bool
2660     #2
2661   } {
2662     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2663       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2664         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2665         \dobrackets { #2 }
2666       }

```

```

2667     }{ #2 }
2668   }
2669 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2670 \bool_new:N \l__stex_terms_brackets_done_bool
2671 %\RequirePackage{scalerel}
2672 \cs_new_protected:Npn \dobrackets #1 {
2673   %\ThisStyle{\if D\m@switch
2674   %   \exp_args:Nnx \use:nn
2675   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2676   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2677   % \else
2678   \exp_args:Nnx \use:nn
2679   {
2680     \bool_set_true:N \l__stex_terms_brackets_done_bool
2681     \int_set:Nn \l__stex_terms_downprec \infprec
2682     \l__stex_terms_left_bracket_str
2683     #1
2684   }
2685   {
2686     \bool_set_false:N \l__stex_terms_brackets_done_bool
2687     \l__stex_terms_right_bracket_str
2688     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2689   }
2690   %\fi}
2691 }

```

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

`\withbrackets`

```

2692 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2693   \exp_args:Nnx \use:nn
2694   {
2695     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2696     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2697     #3
2698   }
2699   {
2700     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2701     {\l__stex_terms_left_bracket_str}
2702     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2703     {\l__stex_terms_right_bracket_str}
2704   }
2705 }

```

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

`\STEXinvisible`

```

2706 \cs_new_protected:Npn \STEXinvisible #1 {
2707   \stex_annotate_invisible:n { #1 }
2708 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2709 \cs_new_protected:Nn \_stex_term_oms:nnn {
2710   \stex_annotate:nnn{ OMID }{ #2 }{
2711     \stex_highlight_term:nn { #1 } { #3 }
2712   }
2713 }
2714
2715 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2716   \__stex_terms_maybe_brackets:nn { #3 }{
2717     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2718   }
2719 }

```

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

`_stex_term_math_oma:nnnn`

```

2720 \cs_new_protected:Nn \_stex_term_oma:nnn {
2721   \stex_annotate:nnn{ OMA }{ #2 }{
2722     \stex_highlight_term:nn { #1 } { #3 }
2723   }
2724 }
2725
2726 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2727   \__stex_terms_maybe_brackets:nn { #3 }{
2728     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2729   }
2730 }

```

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

`_stex_term_math_omb:nnnn`

```

2731 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2732   \stex_annotate:nnn{ OMBIND }{ #2 }{
2733     \stex_highlight_term:nn { #1 } { #3 }
2734   }
2735 }
2736
2737 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2738   \__stex_terms_maybe_brackets:nn { #3 }{
2739     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2740   }
2741 }

```

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

`_stex_term_math_arg:nnn`

```

2742 \cs_new_protected:Nn \_stex_term_arg:nn {
2743   \stex_unhighlight_term:n {
2744     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2745   }
2746 }

```

```

2747 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2748   \exp_args:Nnx \use:nn
2749     { \int_set:Nn \l__stex_terms_downprec { #2 }
2750       \stex_term_arg:nn { #1 }{ #3 }
2751     }
2752     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2753   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2754 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2755   \clist_set:Nn \l_tmpa_clist{ #4 }
2756   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2757     \tl_set:Nn \l_tmpa_tl { #4 }
2758   }{
2759     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2760     \clist_reverse:N \l_tmpa_clist
2761     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2762
2763     \clist_map_inline:Nn \l_tmpa_clist {
2764       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2765         \exp_args:Nno
2766           \l_tmpa_cs { ##1 } \l_tmpa_tl
2767       }
2768     }
2769
2770   }
2771   \exp_args:Nnno
2772   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2773 }

```

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

`\stex_term_custom:nn`

```

2774 \cs_new_protected:Nn \stex_term_custom:nn {
2775   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2776   \str_set:Nn \l_tmpa_str { #2 }
2777   \tl_clear:N \l_tmpa_tl
2778   \int_zero:N \l_tmpa_int
2779   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2780   \__stex_terms_custom_loop:
2781 }

```

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

`__stex_terms_custom_loop:`

```

2782 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2783   \bool_set_false:N \l_tmpa_bool
2784   \bool_while_do:nn {
2785     \str_if_eq_p:ee X {
2786       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2787     }
2788   }{
2789     \int_incr:N \l_tmpa_int

```

```

2790 }
2791
2792 \peek_charcode:NTF [ {
2793   % notation/text component
2794   \__stex_terms_custom_component:w
2795 } {
2796   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2797     % all arguments read => finish
2798     \__stex_terms_custom_final:
2799   } {
2800     % arguments missing
2801     \peek_charcode_remove:NTF * {
2802       % invisible, specific argument position or both
2803       \peek_charcode:NTF [ {
2804         % visible specific argument position
2805         \__stex_terms_custom_arg:wn
2806       } {
2807         % invisible
2808         \peek_charcode_remove:NTF * {
2809           % invisible specific argument position
2810           \__stex_terms_custom_arg_inv:wn
2811         } {
2812           % invisible next argument
2813           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2814         }
2815       }
2816     } {
2817       % next normal argument
2818       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2819     }
2820   }
2821 }
2822 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2823 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2824   \bool_set_true:N \l_tmpa_bool
2825   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2826 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2827 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2828   \str_set:Nx \l_tmpb_str {
2829     \str_item:Nn \l_tmpa_str { #1 }
2830   }
2831   \str_case:VnTF \l_tmpb_str {
2832     { X } {
2833       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2834     }
2835     { i } { \__stex_terms_custom_set_X:n { #1 } }
2836     { b } { \__stex_terms_custom_set_X:n { #1 } }

```



```

2837 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2838 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2839 }{}{
2840 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2841 }
2842
2843 \bool_if:nTF \l_tmpa_bool {
2844   \tl_put_right:Nx \l_tmpa_tl {
2845     \stex_annotate_invisible:n {
2846       \stex_term_arg:nn { \int_eval:n { #1 } }
2847       \exp_not:n { { #2 } }
2848     }
2849   }
2850 } {
2851   \tl_put_right:Nx \l_tmpa_tl {
2852     \stex_term_arg:nn { \int_eval:n { #1 } }
2853     \exp_not:n { { #2 } }
2854   }
2855 }
2856
2857 \_stex_terms_custom_loop:
2858 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2859 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2860   \str_set:Nx \l_tmpa_str {
2861     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2862     X
2863     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2864   }
2865 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2866 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2867   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2868   \_stex_terms_custom_loop:
2869 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2870 \cs_new_protected:Nn \_stex_terms_custom_final: {
2871   \int_compare:nNnTF \l_tmpb_int = 0 {
2872     \exp_args:Nnno \stex_term_oms:nnn
2873   }{
2874     \str_if_in:NnTF \l_tmpa_str {b} {
2875       \exp_args:Nnno \stex_term_ombind:nnn
2876     } {
2877       \exp_args:Nnno \stex_term_oma:nnn
2878     }
2879   }

```

```

2880 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2881 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

2882 \NewDocumentCommand \symref { m m }{
2883   \let\compemph_uri_prev:\compemph@uri
2884   \let\compemph@uri\symrefemph@uri
2885   \STEXsymbol{#1}! [#2]
2886   \let\compemph@uri\compemph_uri_prev:
2887 }
2888
2889 \keys_define:nn { stex / symname } {
2890   post      .str_set_x:N    = \l_stex_symname_post_str
2891 }
2892
2893 \cs_new_protected:Nn \stex_symname_args:n {
2894   \str_clear:N \l_stex_symname_post_str
2895   \keys_set:nn { stex / symname } { #1 }
2896 }
2897
2898 \NewDocumentCommand \symname { 0{} m }{
2899   \stex_symname_args:n { #1 }
2900   \stex_get_symbol:n { #2 }
2901   \str_set:Nx \l_tmpa_str {
2902     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2903   }
2904   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2905
2906   \let\compemph_uri_prev:\compemph@uri
2907   \let\compemph@uri\symrefemph@uri
2908   \exp_args:NNx \use:nn
2909   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2910     \l_tmpa_str \l_stex_symname_post_str
2911   ] }
2912   \let\compemph@uri\compemph_uri_prev:
2913 }

```

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

31.3 Notation Components

```

2914 \<@@=stex_notationcomps>

```

\stex_highlight_term:nn

```

2915
2916 \str_new:N \l_stex_current_symbol_str
2917 \cs_new_protected:Nn \stex_highlight_term:nn {
2918   \exp_args:Nnx
2919   \use:nn {
2920     \str_set:Nx \l_stex_current_symbol_str { #1 }
2921     #2
2922   } {

```

```

2923 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2924 { \l_stex_current_symbol_str }
2925 }
2926 }
2927
2928 \cs_new_protected:Nn \stex_unhighlight_term:n {
2929 % \latexml_if:TF {
2930 % #1
2931 % } {
2932 % \rustex_if:TF {
2933 % #1
2934 % } {
2935 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2936 % }
2937 % }
2938 }

```

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

```

\comp
\compemph@uri 2939 \cs_new_protected:Npn \comp #1 {
\compemph 2940 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 2941 \rustex_if:TF {
\defemph@uri 2942 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 2943 }{
\symrefemph@uri 2944 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2945 }
2946 }
2947 }
2948
2949 \cs_new_protected:Npn \compemph@uri #1 #2 {
2950 \compemph{ #1 }
2951 }
2952
2953
2954 \cs_new_protected:Npn \compemph #1 {
2955 #1
2956 }
2957
2958 \cs_new_protected:Npn \defemph@uri #1 #2 {
2959 \defemph{#1}
2960 }
2961
2962 \cs_new_protected:Npn \defemph #1 {
2963 \textbf{#1}
2964 }
2965
2966 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2967 \symrefemph{#1}
2968 }
2969
2970 \cs_new_protected:Npn \symrefemph #1 {
2971 \textbf{#1}
2972 }

```

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

`\ellipses`

```
2973 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2974 \bool_new:N \l_stex_inarray_bool
\parrayline 2975 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2976 \NewDocumentCommand \parray { m m } {
\parraycell 2977 \begingroup
2978 \bool_set_true:N \l_stex_inarray_bool
2979 \begin{array}{#1}
2980 #2
2981 \end{array}
2982 \endgroup
2983 }
2984
2985 \NewDocumentCommand \prmatrix { m } {
2986 \begingroup
2987 \bool_set_true:N \l_stex_inarray_bool
2988 \begin{matrix}
2989 #1
2990 \end{matrix}
2991 \endgroup
2992 }
2993
2994 \def \maybepline {
2995 \bool_if:NT \l_stex_inarray_bool {\hline}
2996 }
2997
2998 \def \parrayline #1 #2 {
2999 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3000 }
3001
3002 \def \pmrow #1 { \parrayline{}{ #1 } }
3003
3004 \def \parraylineh #1 #2 {
3005 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3006 }
3007
3008 \def \parraycell #1 {
3009 #1 \bool_if:NT \l_stex_inarray_bool {&}
3010 }

```

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

```
3011 \endpackage
```

Chapter 32

STEX -Structural Features Implementation

```
3012 <*package>
3013
3014 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3015
3016 <@@=stex_features>
    Warnings and error messages
3017
```

32.1 Imports with modification

```
3018 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3019   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3020     \__stex_features_get_symbol_from_cs:n { #1 }
3021   }{
3022     % argument is a string
3023     % is it a command name?
3024     \cs_if_exist:cTF { #1 }{
3025       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3026       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3027       \str_if_empty:NTF \l_tmpa_str {
3028         \exp_args:Nx \cs_if_eq:NNTF {
3029           \tl_head:N \l_tmpa_tl
3030         } \stex_invoke_symbol:n {
3031           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3032         }{
3033           \__stex_features_get_symbol_from_string:n { #1 }
3034         }
3035       } {
3036         \__stex_features_get_symbol_from_string:n { #1 }
3037       }
3038     }{
3039       % argument is not a command name
```

```

3040     \__stex_features_get_symbol_from_string:n { #1 }
3041     % \l_stex_all_symbols_seq
3042   }
3043 }
3044 }
3045
3046 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3047   \str_set:Nn \l_tmpa_str { #1 }
3048   \bool_set_false:N \l_tmpa_bool
3049   \bool_if:NF \l_tmpa_bool {
3050     \tl_set:Nn \l_tmpa_tl {
3051       \msg_set:nnn{stex}{error/unknownsymbol}{
3052         No~symbol~#1~found!
3053       }
3054       \msg_error:nn{stex}{error/unknownsymbol}
3055     }
3056     \str_set:Nn \l_tmpa_str { #1 }
3057     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3058     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3059       \str_set:Nn \l_tmpb_str { ##1 }
3060       \str_if_eq:eeT { \l_tmpa_str } {
3061         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3062       } {
3063         \seq_map_break:n {
3064           \tl_set:Nn \l_tmpa_tl {
3065             \str_set:Nn \l_stex_get_symbol_uri_str {
3066               ##1
3067             }
3068             \__stex_features_get_symbol_check:
3069           }
3070         }
3071       }
3072     }
3073     \l_tmpa_tl
3074   }
3075 }
3076
3077 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3078   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3079   { \tl_tail:N \l_tmpa_tl }
3080   \tl_if_single:NTF \l_tmpa_tl {
3081     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3082       \exp_after:wN \str_set:Nn \exp_after:wN
3083       \l_stex_get_symbol_uri_str \l_tmpa_tl
3084       \__stex_features_get_symbol_check:
3085     }{
3086       % TODO
3087       % tail is not a single group
3088     }
3089   }{
3090     % TODO
3091     % tail is not a single group
3092   }
3093 }

```

```

3094
3095 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3096   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3097   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3098     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3099     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3100     \seq_if_in:Nof \l__stex_features_copymodule_modules_seq {
3101       % TODO error
3102     }
3103   }{
3104     % TODO error
3105   }
3106 }
3107
3108 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3109   \stex_import_module_uri:nn { #1 } { #2 }
3110   \stex_deactivate_macro:Nn \symdecl {module~environments}
3111   \stex_deactivate_macro:Nn \symdef {module~environments}
3112   \stex_deactivate_macro:Nn \notation {module~environments}
3113   \stex_reactivate_macro:N \assign
3114   \stex_reactivate_macro:N \renamedec1
3115   \stex_reactivate_macro:N \donotcopy
3116   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3117   %\let\notation\notation_in_copymodules:
3118   %\stex_module_setup:nn {}{ #3 }
3119   \stex_import_require_module:nnnn
3120     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3121     { \l_stex_import_path_str } { \l_stex_import_name_str }
3122   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3123   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3124   \seq_clear:N \l__stex_features_copymodule_fields_seq
3125   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3126     \seq_map_inline:cn {c_stex_module_###1_constants}{
3127       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3128         ##1 ? #####1
3129       }
3130     }
3131   }
3132   \seq_clear:N \l_tmpa_seq
3133   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3134     name      = \l_stex_current_copymodule_name_str ,
3135     module    = \l_stex_current_module_str ,
3136     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3137     includes  = \l_tmpa_seq ,
3138     fields    = \l_tmpa_seq
3139   }
3140   \stex_debug:nn{copymodule}{cloning~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3141     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3142   \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3143   % todo
3144
3145   \stex_if_smsmode:TF {
3146     \stex_smsmode_set_codes:
3147   } {

```

```

3148     \begin{stex_annotate_env} {structure} {
3149         \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3150     }
3151     \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3152 }
3153 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3154 \bool_set_false:N \l_stex_html_do_output_bool
3155 }{
3156     \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3157     \tl_clear:N \l_tmpa_tl
3158     \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3159     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3160         \seq_map_inline:cn {c_stex_module_###1_constants}{\stex_annotate:nnn{assignment} {###1?###1
3161             \str_if_exist:cTF {l__stex_features_copymodule_###1?###1_name_str} {
3162                 \tl_put_right:Nx \l_tmpa_tl {
3163                     \prop_set_from_keyval:cn {
3164                         l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3165                     }{
3166                         \exp_after:wN \prop_to_keyval:N \csname
3167                             l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3168                         \endcsname
3169                     }
3170                     \seq_clear:c {
3171                         l_stex_symdecl_
3172                         \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3173                         _notations
3174                     }
3175                 }
3176             }
3177             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_###1?###1_name_str}}
3178             \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}}
3179             \str_if_exist:cT {l__stex_features_copymodule_###1?###1_macroname_str} {
3180                 \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_###1?###1_name_str}}
3181                 \tl_put_right:Nx \l_tmpa_tl {
3182                     \tl_set:cx {\use:c{l__stex_features_copymodule_###1?###1_macroname_str}}{\}
3183                     \stex_invoke_symbol:n {
3184                         \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3185                     }
3186                 }
3187             }
3188         }
3189     }
3190     \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ###1?###1 _prop}
3191     \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ###1 }
3192     \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3193     \tl_put_right:Nx \l_tmpa_tl {
3194         \prop_set_from_keyval:cn {
3195             l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3196         }{
3197             \prop_to_keyval:N \l_tmpa_prop
3198         }
3199         \seq_clear:c {
3200             l_stex_symdecl_
3201             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ###1
3202             _notations

```



```

3202     }
3203   }
3204   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule
3205   \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3206     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3207     \tl_put_right:Nx \l_tmpa_tl {
3208       \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3209       \stex_invoke_symbol:n {
3210         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3211       }
3212     }
3213   }
3214 }
3215 }
3216 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3217   \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?###
3218 }
3219 % todo notations
3220 }}
3221 }
3222 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3223 \tl_put_left:Nx \l_tmpa_tl {
3224   \prop_set_from_keyval:cn {
3225     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3226   }{
3227     \prop_to_keyval:N \l_stex_current_copymodule_prop
3228   }
3229 }
3230 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3231 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3232 \exp_args:Nx \stex_do_aftergroup:n { \stex_if_smsmode:TF {
3233   \exp_args:No \exp_not:n \l_tmpa_tl
3234   }{ \stex_do_aftergroup:n {
3235     \exp_args:No \exp_not:n \l_tmpa_tl
3236   } }
3237 }
3238 \stex_if_smsmode:F {
3239   \end{stex_annotate_env}
3240 }
3241 }
3242
3243 \NewDocumentCommand \donotcopy { 0{} m}{
3244   \stex_import_module_uri:nn { #1 } { #2 }
3245   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3246   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3247     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3248     \seq_map_inline:cn {c_stex_module_##1_constants}{
3249       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? ####1 }
3250       \bool_lazy_any_p:nT {
3251         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3252         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3253         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3254       }{
3255         % TODO throw error

```

```

3256     }
3257   }
3258 }
3259
3260 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3261 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3262 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3263 }
3264
3265 \NewDocumentCommand \assign { m m }{
3266   \stex_get_symbol_in_copymodule:n {#1}
3267   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3268   \tl_set:cn {l__stex_features_copymodule_##1?####1_def_tl}{#2}
3269 }
3270
3271 \keys_define:nn { stex / renamedec1 } {
3272   name          .str_set_x:N = \l_stex_renamedec1_name_str
3273 }
3274 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3275   \str_clear:N \l_stex_renamedec1_name_str
3276
3277   \keys_set:nn { stex / renamedec1 } { #1 }
3278 }
3279
3280 \NewDocumentCommand \renamedec1 { 0{} m m }{
3281   \__stex_features_renamedec1_args:n { #1 }
3282   \stex_get_symbol_in_copymodule:n {#2}
3283   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3284   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3285   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3286     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3287       \l_stex_get_symbol_uri_str
3288     } }
3289   } {
3290     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3291     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3292     \prop_set_eq:cc {l_stex_symdecl_
3293       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3294       _prop
3295     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3296     \seq_set_eq:cc {l_stex_symdecl_
3297       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3298       _notations
3299     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3300     \prop_put:cnx {l_stex_symdecl_
3301       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3302       _prop
3303     }{ name }{ \l_stex_renamedec1_name_str }
3304     \prop_put:cnx {l_stex_symdecl_
3305       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3306       _prop
3307     }{ module }{ \l_stex_current_module_str }
3308     \exp_args:NNx \seq_put_left:Nn \__stex_features_copymodule_fields_seq {
3309       \l_stex_current_module_str ? \l_stex_renamedec1_name_str

```

```

3310 }
3311 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3312   \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3313 } }
3314 }
3315 }
3316 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3317 % \_stex_notation_args:n { #1 }
3318 % \tl_clear:N \l_stex_symdecl_definiens_tl
3319 % \stex_get_symbol_in_copymodule:n { #2 }
3320 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3321 % % todo
3322 %}
3323 \stex_deactivate_macro:Nn \assign {copymodules}
3324 \stex_deactivate_macro:Nn \renameddecl {copymodules}
3325 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3326
3327
3328 \seq_new:N \l_stex_implicit_morphisms_seq
3329 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3330   \stex_import_module_uri:nn { #1 } { #2 }
3331   \stex_debug:nn{implicits}{
3332     Implicit~morphism:~
3333     \l_stex_module_ns_str ? \l__stex_features_name_str
3334   }
3335   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3336     \l_stex_module_ns_str ? \l__stex_features_name_str
3337   }{
3338     \msg_error:nnn{stex}{error/conflictingmodules}{
3339       \l_stex_module_ns_str ? \l__stex_features_name_str
3340     }
3341   }
3342
3343   % TODO
3344
3345
3346
3347   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3348     \l_stex_module_ns_str ? \l__stex_features_name_str
3349   }
3350 }
3351

```

32.2 The feature environment

structural@feature

```

3352
3353 \NewDocumentEnvironment{structural@feature}{ m m m }{
3354   \stex_if_in_module:F {
3355     \msg_set:nnn{stex}{error/nomodule}{
3356       Structural~Feature~has~to~occur~in~a~module:\\
3357       Feature~#2~of~type~#1\\
3358       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

3359     }
3360     \msg_error:nn{stex}{error/nomodule}
3361 }
3362
3363 \str_set:Nx \l_stex_module_name_str {
3364     \prop_item:Nn \l_stex_current_module_prop
3365     { name } / #2 - feature
3366 }
3367
3368 \str_set:Nx \l_stex_module_ns_str {
3369     \prop_item:Nn \l_stex_current_module_prop
3370     { ns }
3371 }
3372
3373
3374 \str_clear:N \l_tmpa_str
3375 \seq_clear:N \l_tmpa_seq
3376 \tl_clear:N \l_tmpa_tl
3377 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3378     origname = #2,
3379     name     = \l_stex_module_name_str ,
3380     ns       = \l_stex_module_ns_str ,
3381     imports  = \exp_not:o { \l_tmpa_seq } ,
3382     constants = \exp_not:o { \l_tmpa_seq } ,
3383     content  = \exp_not:o { \l_tmpa_tl } ,
3384     file     = \exp_not:o { \g_stex_currentfile_seq } ,
3385     lang     = \l_stex_module_lang_str ,
3386     sig      = \l_tmpa_str ,
3387     meta     = \l_tmpa_str ,
3388     feature  = #1 ,
3389 }
3390
3391 \stex_if_smsmode:TF {
3392     \stex_smsmode_set_codes:
3393 } {
3394     \begin{stex_annotate_env}{ feature:#1 }{}
3395     \stex_annotate_invisible:nnn{header}{}{ #3 }
3396 }
3397 }{
3398     \str_set:Nx \l_tmpa_str {
3399         c_stex_feature_
3400         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3401         \prop_item:Nn \l_stex_current_module_prop { name }
3402         _prop
3403     }
3404     \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3405     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3406     \stex_if_smsmode:TF {
3407         \exp_args:Nx \stex_add_to_sms:n {
3408             \prop_gset_from_keyval:cn {
3409                 c_stex_feature_
3410                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3411                 \prop_item:Nn \l_stex_current_module_prop { name }
3412                 _prop

```

```

3413     } {
3414         origname = #2,
3415         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3416         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3417         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3418         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3419         content   = \prop_item:cn { \l_tmpa_str } { content } ,
3420         file      = \prop_item:cn { \l_tmpa_str } { file } ,
3421         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
3422         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
3423         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
3424         feature   = \prop_item:cn { \l_tmpa_str } { feature }
3425     }
3426 }
3427 } {
3428     \end{stex_annotate_env}
3429 }
3430 }
3431

```

32.3 Features

structure

```

3432
3433 \prop_new:N \l_stex_all_structures_prop
3434
3435 \keys_define:nn { stex / features / structure } {
3436     name .str_set_x:N = \l__stex_features_structure_name_str ,
3437 }
3438
3439 \cs_new_protected:Nn \__stex_features_structure_args:n {
3440     \str_clear:N \l__stex_features_structure_name_str
3441     \keys_set:nn { stex / features / structure } { #1 }
3442 }
3443
3444 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3445 %   \__stex_features_structure_args:n { ##1 }
3446 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3447 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3448 %   }
3449 %} {
3450 %
3451 %}
3452
3453 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3454     \__stex_features_structure_args:n { #1 }
3455     \str_if_empty:NT \l__stex_features_structure_name_str {
3456         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3457     }
3458     \exp_args:Nnnx
3459     \begin{structural@feature}{ structure }
3460         { \l__stex_features_structure_name_str }{}
3461         \seq_clear:N \l_tmpa_seq

```

```

3462 \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3463
3464 }{
3465 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3466 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3467 \str_set:Nx \l_tmpa_str {
3468 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3469 \prop_item:Nn \l_stex_current_module_prop { name }
3470 }
3471 \seq_map_inline:Nn \l_tmpa_seq {
3472 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3473 }
3474 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3475 \exp_args:Nnx
3476 \AddToHookNext { env / mathstructure / after }{
3477 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3478 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3479 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3480 \STEXexport {
3481 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3482 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3483 {\l_tmpa_str}
3484 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3485 {#2}{\l_tmpa_str}
3486 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3487 % \prop_item:Nn \l_stex_current_module_prop { origname },
3488 % \l_tmpa_str
3489 % }
3490 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3491 % #2,\l_tmpa_str
3492 % }
3493 % \tl_set:cx { #2 } {
3494 % \stex_invoke_structure:n { \l_tmpa_str }
3495 % }
3496 % }
3497
3498 \end{structural@feature}
3499 % \g_stex_last_feature_prop
3500 }

```

\instantiate

```

3501 \seq_new:N \l__stex_features_structure_field_seq
3502 \str_new:N \l__stex_features_structure_field_str
3503 \str_new:N \l__stex_features_structure_def_tl
3504 \prop_new:N \l__stex_features_structure_prop
3505 \NewDocumentCommand \instantiate { m O{} m }{
3506 \stex_smsmode_set_codes:
3507 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3508 \prop_set_eq:Nc \l__stex_features_structure_prop {
3509 c_stex_feature_\l_tmpa_str _prop
3510 }
3511 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3512 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3513 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }

```

```

3514 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3515   \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3516   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3517     {} \l_tmpa_tl
3518   \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3519     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3520     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3521     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3522   }{
3523     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3524     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3525     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3526       \l_tmpa_tl
3527     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3528       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3529       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3530     }{
3531       \tl_clear:N \l_tmpb_tl
3532     }
3533   }
3534 }{
3535   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3536   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3537     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3538     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3539     \tl_clear:N \l_tmpa_tl
3540   }{
3541     % TODO throw error
3542   }
3543 }
3544 % \l_tmpa_str: name
3545 % \l_tmpa_tl: definiens
3546 % \l_tmpb_tl: notation
3547 \tl_if_empty:NT \l__stex_features_structure_field_str {
3548   % TODO throw error
3549 }
3550 \str_clear:N \l_tmpb_str
3551
3552 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3553 \seq_map_inline:Nn \l_tmpa_seq {
3554   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3555   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3556   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3557     \seq_map_break:n {
3558       \str_set:Nn \l_tmpb_str { ####1 }
3559     }
3560   }
3561 }
3562 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3563   \l_tmpb_str
3564
3565 \tl_if_empty:NNTF \l_tmpb_tl {
3566   \tl_if_empty:NF \l_tmpa_tl {
3567     \exp_args:Nx \use:n {

```

```

3568         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3569     }
3570 }
3571 }{
3572     \tl_if_empty:NTF \l_tmpa_tl {
3573         \exp_args:Nx \use:n {
3574             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3575         }
3576     }
3577 }{
3578     \exp_args:Nx \use:n {
3579         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3580         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3581     }
3582 }
3583 }
3584 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3585 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3586 % #3/\l__stex_features_structure_field_str
3587 % \par
3588 % \expandafter\present\csname
3589 %     l_stex_symdecl_
3590 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
3591 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3592 % #3/\l__stex_features_structure_field_str
3593 % _prop
3594 % \endcsname
3595 }
3596
3597 \tl_clear:N \l__stex_features_structure_def_tl
3598
3599 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3600 \seq_map_inline:Nn \l_tmpa_seq {
3601     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3602     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3603     \exp_args:Nx \use:n {
3604         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3605
3606         }
3607     }
3608 }
3609
3610 \prop_if_exist:cF {
3611     l_stex_symdecl_
3612     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3613     \prop_item:Nn \l_stex_current_module_prop {name} ?
3614     #3/\l_tmpa_str
3615     _prop
3616 }{
3617     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3618     \l_tmpb_str
3619     \exp_args:Nx \use:n {
3620         \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3621     }
3622 }

```



```

3622 }
3623
3624 \symdecl*[type={\STEXsymbol{module-type}}{
3625   \stex_term_math_oms:nnnn {
3626     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3627     \prop_item:Nn \l__stex_features_structure_prop {name}
3628     }{}{0}{}
3629   }{}{#3}
3630
3631   % TODO: -> sms file
3632
3633   \tl_set:cx{ #3 }{
3634     \stex_invoke_structure:nnn {
3635       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3636       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3637     } {
3638       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3639       \prop_item:Nn \l__stex_features_structure_prop {name}
3640     }
3641   }
3642
3643 }

```

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

\stex_invoke_structure:nnn

```

3644 % #1: URI of the instance
3645 % #2: URI of the instantiated module
3646 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3647   \tl_if_empty:nTF{ #3 }{
3648     \prop_set_eq:Nc \l__stex_features_structure_prop {
3649       c_stex_feature_ #2 _prop
3650     }
3651     \tl_clear:N \l_tmpa_tl
3652     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3653     \seq_map_inline:Nn \l_tmpa_seq {
3654       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3655       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3656       \cs_if_exist:cT {
3657         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3658       }{
3659         \tl_if_empty:NF \l_tmpa_tl {
3660           \tl_put_right:Nn \l_tmpa_tl {,}
3661         }
3662         \tl_put_right:Nx \l_tmpa_tl {
3663           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3664         }
3665       }
3666     }
3667     \exp_args:No \mathstruct \l_tmpa_tl
3668   }{
3669     \stex_invoke_symbol:n{#1/#3}
3670   }
3671 }

```

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

3672 `\endpackage`

Chapter 33

STEX -Statements Implementation

```
3673 <*package>
3674
3675 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3676
3677 \protected\def\ignorespacesandpars{
3678   \begingroup\catcode13=10\relax
3679   \@ifnextchar\par{
3680     \endgroup\expandafter\ignorespacesandpars\@gobble
3681   }{
3682     \endgroup
3683   }
3684 }
3685
3686 <@@=stex_statements>
3687
3688   Warnings and error messages
```

\titleemph

```
3688 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3689 \keys_define:nn {stex / definiendum }{
3690   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3691   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3692   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3693 }
3694 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3695   \str_clear:N \l__stex_statements_definiendum_root_str
3696   \tl_clear:N \l__stex_statements_definiendum_post_tl
3697   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3698 \keys_set:nn { stex / definiendum }{ #1 }
3699 }
3700 \NewDocumentCommand \definiendum { 0{ } m m } {
3701   \__stex_statements_definiendum_args:n { #1 }
3702   \stex_get_symbol:n { #2 }
3703   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3704   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3705     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3706       \tl_set:Nn \l_tmpa_tl { #3 }
3707     } {
3708       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3709       \tl_set:Nn \l_tmpa_tl {
3710         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3711       }
3712     }
3713   } {
3714     \tl_set:Nn \l_tmpa_tl { #3 }
3715   }
3716
3717   % TODO root
3718   \rustex_if:TF {
3719     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3720   } {
3721     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3722   }
3723 }
3724 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3725 \NewDocumentCommand \definame { 0{ } m } {
3726   \__stex_statements_definiendum_args:n { #1 }
3727   % TODO: root
3728   \stex_get_symbol:n { #2 }
3729   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3730   \str_set:Nx \l_tmpa_str {
3731     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3732   }
3733   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3734   \rustex_if:TF {
3735     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3736       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3737     }
3738   } {
3739     \defemph@uri {
3740       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3741     } { \l_stex_get_symbol_uri_str }
3742   }
3743 }
3744 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3745
3746 \keys_define:nn {stex / sdefinition }{
3747   type      .str_set_x:N = \sdefinitiontype,
3748   id        .str_set_x:N = \sdefinitionid,
3749   title     .tl_set:N     = \sdefinitiontitle
3750 }
3751 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3752   \str_clear:N \sdefinitiontype
3753   \str_clear:N \sdefinitionid
3754   \tl_clear:N \sdefinitiontitle
3755   \keys_set:nn { stex / sdefinition }{ #1 }
3756 }
3757
3758 \NewDocumentEnvironment{sdefinition}{0{}}{
3759   \__stex_statements_sdefinition_args:n{ #1 }
3760   \stex_reactivate_macro:N \definiendum
3761   \stex_reactivate_macro:N \definame
3762   \stex_smsmode_set_codes:
3763   \clist_set:No \l_tmpa_clist \sdefinitiontype
3764   \tl_clear:N \l_tmpa_tl
3765   \clist_map_inline:Nn \l_tmpa_clist {
3766     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3767       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3768     }
3769   }
3770   \stex_if_smsmode:F {
3771     \exp_args:Nnnx
3772     \begin{stex_annotate_env}{definition}{}
3773     \str_if_empty:NF \sdefinitiontype {
3774       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3775     }
3776   }
3777   \tl_if_empty:NTF \l_tmpa_tl {
3778     \__stex_statements_sdefinition_start:
3779   }{
3780     \l_tmpa_tl
3781   }
3782   \stex_ref_new_doc_target:n \sdefinitionid
3783 }{
3784   \clist_set:No \l_tmpa_clist \sdefinitiontype
3785   \tl_clear:N \l_tmpa_tl
3786   \clist_map_inline:Nn \l_tmpa_clist {
3787     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3788       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3789     }
3790   }
3791   \tl_if_empty:NTF \l_tmpa_tl {
3792     \__stex_statements_sdefinition_end:
3793   }{
3794     \l_tmpa_tl
3795   }
3796   \stex_if_smsmode:F {
3797     \end{stex_annotate_env}

```

```

3798 }
3799 }

```

`\stexpatchdefinition`

```

3800 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3801   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3802     ~(\sdefinitiontitle)
3803   }~}
3804 }
3805 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3806
3807 \newcommand\stexpatchdefinition[3] [] {
3808   \str_set:Nx \l_tmpa_str{ #1 }
3809   \str_if_empty:NTF \l_tmpa_str {
3810     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3811     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3812   }{
3813     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3814     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3815   }
3816 }

```

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

`\inlinedef inline:`

```

3817 \NewDocumentCommand \inlinedef { m } {
3818   \beginngroup
3819   \stex_reactivate_macro:N \definiendum
3820   \stex_reactivate_macro:N \definame
3821   \stex_ref_new_doc_target:n{
3822     #1
3823   }
3824 }

```

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

33.2 Assertions

`sassertion`

```

3825
3826 \keys_define:nn {stex / sassertion }{
3827   type      .str_set_x:N = \sassertiontype,
3828   id        .str_set_x:N = \sassertionid,
3829   title     .tl_set:N     = \sassertiontitle ,
3830   name      .str_set_x:N = \sassertionname
3831 }
3832 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3833   \str_clear:N \sassertiontype
3834   \str_clear:N \sassertionid
3835   \str_clear:N \sassertionname
3836   \tl_clear:N \sassertiontitle
3837   \keys_set:nn { stex / sassertion }{ #1 }
3838 }

```

```

3839
3840 %\tl_new:N \g__stex_statements_aftergroup_tl
3841
3842 \NewDocumentEnvironment{sassertion}{0{}}{
3843   \__stex_statements_sassertion_args:n{ #1 }
3844   \stex_smsmode_set_codes:
3845   \clist_set:No \l_tmpa_clist \sassertiontype
3846   \tl_clear:N \l_tmpa_tl
3847   \clist_map_inline:Nn \l_tmpa_clist {
3848     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3849       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3850     }
3851   }
3852   \stex_if_smsmode:F {
3853     \exp_args:Nnnx
3854     \begin{stex_annotate_env}{assertion}{}
3855     \str_if_empty:NF \sassertiontype {
3856       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3857     }
3858   }
3859   \tl_if_empty:NTF \l_tmpa_tl {
3860     \__stex_statements_sassertion_start:
3861   }{
3862     \l_tmpa_tl
3863   }
3864   \stex_ref_new_doc_target:n \sassertionid
3865 }{
3866   \clist_set:No \l_tmpa_clist \sassertiontype
3867   \tl_clear:N \l_tmpa_tl
3868   \clist_map_inline:Nn \l_tmpa_clist {
3869     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3870       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3871     }
3872   }
3873   \tl_if_empty:NTF \l_tmpa_tl {
3874     \__stex_statements_sassertion_end:
3875   }{
3876     \l_tmpa_tl
3877   }
3878   \stex_if_smsmode:F {
3879     \end{stex_annotate_env}
3880   }
3881   % \str_if_empty:NF \sassertionname {
3882   %   \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3883   %     \symdecl*{\sassertionname}
3884   %   }
3885   %   \aftergroup\g__stex_statements_aftergroup_tl
3886   % }
3887 }

```

\stexpatchassertion

```

3888
3889 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3890   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3891      (\sassertiontitle)
3892    }~}
3893  }
3894  \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3895
3896  \newcommand\stexpatchassertion[3] [] {
3897    \str_set:Nx \l_tmpa_str{ #1 }
3898    \str_if_empty:NTF \l_tmpa_str {
3899      \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3900      \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3901    }{
3902      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3903      \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3904    }
3905  }

```

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

\inlineass inline:

```

3906  \NewDocumentCommand \inlineass { m } {
3907    \begingroup
3908    \stex_ref_new_doc_target:n{
3909      #1
3910    \endgroup
3911  }

```

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

33.3 Examples

sexample

```

3912
3913  \keys_define:nn {stex / sexample }{
3914    type      .str_set_x:N = \exampletype,
3915    id        .str_set_x:N = \sexampleid,
3916    title     .tl_set:N = \sexampltitle,
3917    for       .clist_set:N = \sexamplefor,
3918  }
3919  \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3920    \str_clear:N \sexampletype
3921    \str_clear:N \sexampleid
3922    \tl_clear:N \sexampltitle
3923    \clist_clear:N \sexamplefor
3924    \keys_set:nn { stex / sexample }{ #1 }
3925  }
3926
3927  \NewDocumentEnvironment{sexample}{0{}}{
3928    \__stex_statements_sexample_args:n{ #1 }
3929    \stex_smsmode_set_codes:
3930    \clist_set:N \l_tmpa_clist \sexampletype
3931    \tl_clear:N \l_tmpa_tl
3932    \clist_map_inline:Nn \l_tmpa_clist {
3933      \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```



```

3934     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3935   }
3936 }
3937 \stex_if_smsmode:F {
3938   \seq_clear:N \l_tmpa_seq
3939   \clist_map_inline:Nn \sexamplefor {
3940     \str_if_eq:nnF{ ##1 }{ }{
3941       \stex_get_symbol:n { ##1 }
3942       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3943         \l_stex_get_symbol_uri_str
3944       }
3945     }
3946   }
3947   \exp_args:Nnnx
3948   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3949   \str_if_empty:NF \sexamplotype {
3950     \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
3951   }
3952 }
3953 \tl_if_empty:NTF \l_tmpa_tl {
3954   \__stex_statements_sexample_start:
3955 }{
3956   \l_tmpa_tl
3957 }
3958 \stex_ref_new_doc_target:n \sexampleid
3959 }{
3960   \clist_set:Nn \l_tmpa_clist \sexamplotype
3961   \tl_clear:N \l_tmpa_tl
3962   \clist_map_inline:Nn \l_tmpa_clist {
3963     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3964       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3965     }
3966   }
3967   \tl_if_empty:NTF \l_tmpa_tl {
3968     \__stex_statements_sexample_end:
3969   }{
3970     \l_tmpa_tl
3971   }
3972   \stex_if_smsmode:F {
3973     \end{stex_annotate_env}
3974   }
3975 }

```

\stexpatchexample

```

3976
3977 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3978   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3979     (\sexamplotype)
3980   }~}
3981 }
3982 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3983
3984 \newcommand\stexpatchexample[3][ ] {
3985   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3986 \str_if_empty:NTF \l_tmpa_str {
3987   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3988   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3989 }{
3990   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3991   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3992 }
3993 }

```

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

`\inlineex` inline:

```

3994 \NewDocumentCommand \inlineex { m } {
3995   \begingroup
3996   \stex_ref_new_doc_target:n{
3997     #1
3998   }
3999 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4000 \keys_define:nn { stex / sparagraph } {
4001   id      .str_set_x:N = \sparagraphid ,
4002   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4003   type    .str_set_x:N = \sparagraphtype ,
4004   for     .str_set_x:N = \sparagraphfor ,
4005   from    .tl_set_x:N  = \sparagraphfrom ,
4006   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4007   name    .str_set:N   = \sparagraphname
4008 }
4009
4010 \cs_new_protected:Nn \stex_sparagraph_args:n {
4011   \tl_clear:N \l_stex_sparagraph_title_tl
4012   \tl_clear:N \sparagraphfrom
4013   \tl_clear:N \l_stex_sparagraph_start_tl
4014   \str_clear:N \sparagraphid
4015   \str_clear:N \sparagraphtype
4016   \str_clear:N \sparagraphfor
4017   \str_clear:N \sparagraphname
4018   \keys_set:nn { stex / sparagraph } { #1 }
4019 }
4020 \newif\if@in@omtext\@in@omtextfalse
4021
4022 \NewDocumentEnvironment {sparagraph} { 0{} } {
4023   \stex_sparagraph_args:n { #1 }
4024   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4025     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4026   }{
4027     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4028   }

```

```

4029 \@in@omtexttrue
4030 \stex_smsmode_set_codes:
4031 \clist_set:No \l_tmpa_clist \sparagraphtype
4032 \tl_clear:N \l_tmpa_tl
4033 \clist_map_inline:Nn \l_tmpa_clist {
4034   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4035     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4036   }
4037 }
4038 \stex_if_smsmode:F {
4039   \exp_args:Nnnx
4040   \begin{stex_annotate_env}{paragraph}{}
4041   \str_if_empty:NF \sparagraphtype {
4042     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4043   }
4044 }
4045 \tl_if_empty:NTF \l_tmpa_tl {
4046   \__stex_statements_sparagraph_start:
4047 }{
4048   \l_tmpa_tl
4049 }
4050 \stex_ref_new_doc_target:n \sparagraphid
4051 \ignorespacesandpars
4052 }{
4053   \clist_set:No \l_tmpa_clist \sparagraphtype
4054   \tl_clear:N \l_tmpa_tl
4055   \clist_map_inline:Nn \l_tmpa_clist {
4056     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4057       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4058     }
4059   }
4060   \tl_if_empty:NTF \l_tmpa_tl {
4061     \__stex_statements_sparagraph_end:
4062   }{
4063     \l_tmpa_tl
4064   }
4065   \stex_if_smsmode:F {
4066     \end{stex_annotate_env}
4067   }
4068   % \str_if_empty:NF \sparagraphname {
4069   %   \tl_gset:Nx \g__stex_statements_aftergroup_tl {
4070   %     \symdecl*\sparagraphname}
4071   %   }
4072   %   \aftergroup\g__stex_statements_aftergroup_tl
4073   % }
4074 }

```

\stexpatchparagraph

```

4075
4076 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4077   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4078     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4079       \titleemph{\l_stex_sparagraph_title_tl}:~
4080     }

```

```

4081   }{
4082     \titleemph{\l_stex_sparagraph_start_tl}~
4083   }
4084 }
4085 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4086
4087 \newcommand\stexpatchparagraph[3] [] {
4088   \str_set:Nx \l_tmpa_str{ #1 }
4089   \str_if_empty:NTF \l_tmpa_str {
4090     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4091     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4092   }{
4093     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4094     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4095   }
4096 }

```

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

symboldoc

```

4097 \NewDocumentEnvironment{symboldoc}{m}{
4098   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4099   \seq_clear:N \l_tmpb_seq
4100   \seq_map_inline:Nn \l_tmpa_seq {
4101     \str_if_eq:nnF{ ##1 }{}{
4102       \stex_get_symbol:n { ##1 }
4103       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4104         \l_stex_get_symbol_uri_str
4105       }
4106     }
4107   }
4108   \par
4109   \exp_args:Nnnx
4110   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4111 }{
4112   \end{stex_annotate_env}
4113 }
4114 \</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

```
4115 <*package>
4116 <@@=stex_sproof>
4117
4118 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4119
```

34.2 Proofs

We first define some keys for the proof environment.

```
4120 \keys_define:nn { stex / spf } {
4121   id          .str_set:N = \l__stex_sproof_spf_id_str,
4122   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4123   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4124   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4125   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4126   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4127   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4128   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4129   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4130   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4131 }
4132 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4133   \str_clear:N \l__stex_sproof_spf_id_str
4134   \tl_clear:N \l__stex_sproof_spf_display_tl
4135   \tl_clear:N \l__stex_sproof_spf_for_tl
4136   \tl_clear:N \l__stex_sproof_spf_from_tl
4137   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4138   \tl_clear:N \l__stex_sproof_spf_type_tl
4139   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4140 \tl_clear:N \l__stex_sproof_spf_continues_tl
4141 \tl_clear:N \l__stex_sproof_spf_functions_tl
4142 \tl_clear:N \l__stex_sproof_spf_method_tl
4143 \keys_set:nn { stex / spf }{ #1 }
4144 }

```

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

```

4145 \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.

```

4146 \newcount\count_ten
4147 \newenvironment{pst@with@label}[1]{
4148   \edef\pst@label{#1}
4149   \advance\count_ten by 1\relax
4150   \count_ten=1
4151 }{
4152   \advance\count_ten by -1\relax
4153 }

```

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

```

4154 \def\the@pst@label{
4155   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4156 }

```

(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.

```

4157 \keys_define:nn { stex / pstlabel }{
4158   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4159   delimiter    .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4160   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4161 }
4162 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4163 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4164 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4165 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4166 }
4167 \__stex_sproof_pstlabel_args:n {}
4168 \newcommand\setpstlabelstyle[1]{
4169   \__stex_sproof_pstlabel_args:n {#1}
4170 }
4171 \newcommand\setpstlabelstyledefault{%
4172   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4173 }

```

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

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

```

4174 \ExplSyntaxOff
4175 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4176 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4177 \def\pst@make@label@short#1#2{#2}
4178 \def\pst@make@label@empty#1#2{}
4179 \ExplSyntaxOn
4180 \def\pstlabelstyle#1{%
4181   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4182 }%
4183 \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.

```

4184 \def\next@pst@label{%
4185   \global\advance\count\count10 by 1%
4186 }%

```

(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

```

4187 \def\sproof@box{
4188   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4189 }
4190 \def\spf@proofend{\sproof@box}
4191 \def\sproofend{
4192   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4193     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4194   }
4195 }
4196 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4197 \def\spf@proofsketch@kw{Proof Sketch}
4198 \def\spf@proof@kw{Proof}
4199 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4200 \cs_if_exist:NT \bbl@loaded {
4201   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4202   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4203     \input{proof-ngerman.ldf}
4204   }
4205   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4206     \input{proof-finnish.ldf}
4207   }
4208   \clist_if_in:NnT \l_tmpa_clist {french}{
4209     \input{proof-french.ldf}
4210   }
4211   \clist_if_in:NnT \l_tmpa_clist {russian}{
4212     \input{proof-russian.ldf}
4213   }
4214 }
4215

```

spfsketch

```

4216 \newcommand\spfsketch[2][]{
4217   \__stex_sproof_spf_args:n{#1}
4218   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4219     \titleemph{
4220       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4221         \spf@proofsketch@kw
4222       }{
4223         \l__stex_sproof_spf_type_tl
4224       }
4225     }:
4226   }
4227   {-#2}
4228   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4229   \sproofend
4230 }

```

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

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

```

4231 \newenvironment{spfeq}[2][]{
4232   \__stex_sproof_spf_args:n{#1}
4233   %\sref@target
4234   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4235     \titleemph{
4236       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4237         \spf@proof@kw
4238       }{
4239         \l__stex_sproof_spf_type_tl
4240       }
4241     }:

```

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

¹⁵EDNOTE: document above


```

4242 }
4243 {~#2}
4244 \begin{displaymath}\begin{array}{rcll}
4245 }{
4246 \end{array}\end{displaymath}
4247 }

```

(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.

```

4248 \newenvironment{spf@proof}[2][]{
4249 \__stex_sproof_spf_args:n{#1}
4250 %\sref@target
4251 \count_ten=10
4252 \par\noindent
4253 \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4254 \titleemph{
4255 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4256 \spf@proof@kw
4257 }{
4258 \l__stex_sproof_spf_type_tl
4259 }
4260 } :
4261 }
4262 {~#2}
4263 %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4264 \def\pst@label{}
4265 \newcount\pst@count% initialize the labeling mechanism
4266 \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4267 }{
4268 \end{pst@with@label}\end{description}
4269 }
4270 \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4271 \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidf

```

4272 \newcommand\spfidf[2][]{
4273 \__stex_sproof_spf_args:n{#1}
4274 \titleemph{
4275 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4276 \l__stex_sproof_spf_type_tl
4277 } :
4278 }~#2
4279 \sproofend
4280 }

```

(End definition for `\spfidf`. 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.

spfstep 16

```

4281 \newenvironment{spfstep}[1][]{
4282   \_stex_sproof_spf_args:n{#1}
4283   \@in@omtexttrue
4284   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4285     \item[\the@pst@label]
4286   }
4287   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4288     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4289   }
4290   %\sref@label@id{\pst@label}
4291   \ignorespacesandpars
4292 }{
4293   \next@pst@label\ignorespacesandpars
4294 }

```

sproofcomment

```

4295 \newenvironment{sproofcomment}[1][]{
4296   \_stex_sproof_spf_args:n{#1}
4297   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4298     \item[\the@pst@label]
4299   }
4300 }{
4301   \next@pst@label
4302 }

```

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.

```

4303 \newenvironment{subproof}[2][]{
4304   \_stex_sproof_spf_args:n{#1}
4305   \def\@test{#2}
4306   \ifx\@test\empty\else
4307     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4308       \item[\the@pst@label]
4309     }{#2}
4310   \fi
4311   \begin{pst@with@label}{\pst@label,\number\count_ten}
4312 }{
4313   \end{pst@with@label}\next@pst@label
4314 }

```

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

```

4315 \newenvironment{spfcases}[2][]{
4316   \def\@test{#1}
4317   \ifx\@test\empty
4318     \begin{subproof}[method=by-cases]{#2}
4319   \else
4320     \begin{subproof}[#1,method=by-cases]{#2}
4321   \fi
4322 }{

```

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

```

4323 \end{subproof}
4324 }

```

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

```

4325 \newenvironment{spfcase}[2] [] {
4326   \__stex_sproof_spf_args:n{#1}
4327   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4328     \item[\the@pst@label]
4329   }
4330   \def\@test{#2}
4331   \ifx\@test\@empty
4332   \else
4333     {\titleemph{#2}:~}
4334   \fi
4335   \begin{pst@with@label}{\pst@label,\number\count_ten}
4336 }{
4337   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4338     \sproofend
4339   }
4340   \end{pst@with@label}
4341   \next@pst@label
4342 }

```

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

```

4343 \newcommand\spfcasesketch[3] [] {
4344   \__stex_sproof_spf_args:n{#1}
4345   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4346     \item[\the@pst@label]
4347   }
4348   \def\@test{#2}
4349   \ifx\@test\@empty
4350   \else
4351     {\titleemph{#2}:~}
4352   \fi#3
4353   \next@pst@label
4354 }%

```

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.

```

4355 \keys_define:nn { stex / just }{
4356   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4357   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4358   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4359   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4360 }

```

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

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

justification

```
4361 \newenvironment{justification}[1] [] {}{}
```

\premise

```
4362 \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.

```
4363 \newcommand\justarg[2] [] {#2}
```

```
4364 \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.

Chapter 35

STEX -Others Implementation

```
4365 <*package>
4366
4367 %%%%%%%%%% others.dtx %%%%%%%%%%
4368
4369 <@@=stex_others>
    Warnings and error messages
4370 % None

\MSC Math subject classifier

4371 \NewDocumentCommand \MSC {m} {
4372 % TODO
4373 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4374 \@ifpackageloaded{tikzinput}{
4375 \RequirePackage{stex-tikzinput}
4376 }{}
4377 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4378 \*package>
4379 \@@=stex_modules>
4380
4381 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4382
4383 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4384 \begingroup
4385 \stex_module_setup:nn{
4386   ns=\c_stex_metatheory_ns_str,
4387   meta=NONE
4388 }{Metatheory}
4389 \stex_reactivate_macro:N \symdecl
4390 \stex_reactivate_macro:N \notation
4391 \stex_reactivate_macro:N \symdef
4392 \ExplSyntaxOff
4393 \csname stex_suppress_html:n\endcsname{
4394   % is-a (a:A, a \in A, a is an A, etc.)
4395   \symdecl[args=ai]{isa}
4396   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4397   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4398   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4399
4400   % bind (\forall, \Pi, \lambda etc.)
4401   \symdecl[args=Bi]{bind}
4402   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4403   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4404   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4405
4406   % dummy variable
4407   \symdecl{dummyvar}
4408   \notation[underscore]{dummyvar}{\comp\_}
4409   \notation[dot]{dummyvar}{\comp\cdot}
4410   \notation[dash]{dummyvar}{\comp{\rm --}}
4411
4412   %fromto (function space, Hom-set, implication etc.)
```

```

4413 \symdecl[args=ai]{fromto}
4414 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4415 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4416
4417 % mapto (lambda etc.)
4418 %\symdecl[args=Bi]{mapto}
4419 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4420 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4421 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4422
4423 % function/operator application
4424 \symdecl[args=ia]{apply}
4425 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4426 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4427
4428 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4429 \symdecl{collection}
4430 \notation[U]{collection}{\comp{\mathcal{U}}}
4431 \notation[set]{collection}{\comp{\textsf{Set}}}
4432
4433 % sequences
4434 \symdecl[args=1]{seqtype}
4435 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4436
4437 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4438 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4439
4440 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4441 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4442 % ^ superceded by \aseqfromto and \livar/\uivar
4443
4444 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4445 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4446 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}}{#1\comp,#2}
4447
4448 % letin (‘‘let’’, local definitions, variable substitution)
4449 \symdecl[args=bii]{letin}
4450 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4451 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4452 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4453
4454 % structures
4455 \symdecl*[args=1]{module-type}
4456 \notation{module-type}{\mathtt{MOD} #1}
4457 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4458 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4459
4460 }
4461 \ExplSyntaxOn
4462 \stex_add_to_current_module:n{
4463   \let\nappa\apply
4464   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4465   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4466   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4467 \def\uivar{\csname sequence-index\endcsname[ui]}
4468 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4469 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4470 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4471 }
4472 \__stex_modules_end_module:
4473 \endgroup
4474 \</package>

```


Chapter 37

Tikzinput Implementation

```
4475 <*package>
4476
4477 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4478
4479 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4480 \RequirePackage{l3keys2e}
4481
4482 \keys_define:nn { tikzinput } {
4483   image .bool_set:N = \c_tikzinput_image_bool,
4484   image .default:n = false ,
4485   unknown .code:n = {}
4486 }
4487
4488 \ProcessKeysOptions { tikzinput }
4489
4490 \bool_if:NTF \c_tikzinput_image_bool {
4491   \RequirePackage{graphicx}
4492
4493   \providecommand\usetikzlibrary[]{}
4494   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4495 }{
4496   \RequirePackage{tikz}
4497   \RequirePackage{standalone}
4498
4499   \newcommand \tikzinput [2] [] {
4500     \setkeys{Gin}{#1}
4501     \ifx \Gin@ewidth \Gin@exclamation
4502       \ifx \Gin@eheight \Gin@exclamation
4503         \input { #2 }
4504       \else
4505         \resizebox{!}{ \Gin@eheight }{
4506           \input { #2 }
4507         }
4508       \fi
4509     \else
4510       \ifx \Gin@eheight \Gin@exclamation
4511         \resizebox{ \Gin@ewidth }{!}{
4512           \input { #2 }
```

```

4513     }
4514     \else
4515         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4516             \input { #2 }
4517         }
4518     \fi
4519 \fi
4520 }
4521 }
4522
4523 \newcommand \ctikzinput [2] [] {
4524     \begin{center}
4525         \tikzinput [1] {#2}
4526     \end{center}
4527 }
4528
4529 \@ifpackageloaded{stex}{
4530     \RequirePackage{stex-tikzinput}
4531 }{}
4532
4533 </package>
4534 <*stex>
4535 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4536 \RequirePackage{stex}
4537 \RequirePackage{tikzinput}
4538
4539 \newcommand\mhtikzinput [2] [] {%
4540     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4541     \stex_in_repository:nn\Gin@mhrepos{
4542         \tikzinput [1]{\mhpath{##1}{#2}}
4543     }
4544 }
4545 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4546 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The OMDoc Class

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

```
4547 \*cls)
4548 \@@=document_structure)
4549 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4550 \RequirePackage{l3keys2e,expl-keystr-compat}
```

38.2 Class Options

To initialize the `omdoc` 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`

```
4551 \keys_define:nn{ document-structure / pkg }{
4552   class      .str_set_x:N = \c_document_structure_class_str,
4553   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4554   report     .code:n      = {
4555     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4556     \str_set:Nn \c_document_structure_class_str {report}
4557   },
4558   book       .code:n      = {
4559     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4560     \str_set:Nn \c_document_structure_class_str {book}
4561   },
4562   bookpart   .code:n      = {
4563     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4564     \str_set:Nn \c_document_structure_class_str {book}
4565     \str_set:Nn \c_document_structure_topsect_str {chapter}
4566   },
```

```

4567 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4568 unknown     .code:n      = {
4569   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4570 }
4571 }
4572 \ProcessKeysOptions{ document-structure / pkg }
4573 \str_if_empty:NT \c_document_structure_class_str {
4574   \str_set:Nn \c_document_structure_class_str {article}
4575 }
4576 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4577   {\c_document_structure_class_str}
4578

```

38.3 Beefing up the document environment

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

```

4579 \RequirePackage{omdoc}
4580 \bool_if:NF \c_document_structure_minimal_bool {
4581   \RequirePackage{stex-compatibility}

```

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

```

4582 \keys_define:nn { document-structure / document }{
4583   id .str_set_x:N = \c_document_structure_document_id_str
4584 }
4585 \let\__document_structure_orig_document=\document
4586 \renewcommand{\document}[1][]{
4587   \keys_set:nn{ document-structure / document }{ #1 }
4588   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4589   \__document_structure_orig_document
4590 }

```

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

```

4591 }
4592 \</cls>

```

38.4 Implementation: OMDoc Package

```

4593 \*package>
4594 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4595 \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

```

4596
4597 \keys_define:nn{ document-structure / pkg }{
4598   class      .str_set_x:N = \c_document_structure_class_str,
4599   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4600   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4601 }
4602 \ProcessKeysOptions{ document-structure / pkg }
4603 \str_if_empty:NT \c_document_structure_class_str {
4604   \str_set:Nn \c_document_structure_class_str {article}
4605 }
4606 \str_if_empty:NT \c_document_structure_topsect_str {
4607   \str_set:Nn \c_document_structure_topsect_str {section}
4608 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```

4609 \RequirePackage{xspace}
4610 \RequirePackage{comment}
4611 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

We set up triggers for the other languages, currently only German.

```

4612 \@ifpackageloaded{babel}{
4613   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4614   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4615     \input{omdoc-ngerman.ldf}
4616   }
4617 }{}
4618 %\AfterBabelLanguage{ngerman}{\input{omdoc-ngerman.ldf}}

```

`\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.

```

4619 \int_new:N \l_document_structure_section_level_int
4620 \str_case:VnF \c_document_structure_topsect_str {
4621   {part}{
4622     \int_set:Nn \l_document_structure_section_level_int {0}
4623   }
4624   {chapter}{
4625     \int_set:Nn \l_document_structure_section_level_int {1}
4626   }
4627 }{
4628   \str_case:VnF \c_document_structure_class_str {
4629     {book}{
4630       \int_set:Nn \l_document_structure_section_level_int {0}
4631     }
4632     {report}{
4633       \int_set:Nn \l_document_structure_section_level_int {0}
4634     }
4635   }{
4636     \int_set:Nn \l_document_structure_section_level_int {2}
4637   }
4638 }

```

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

```
4639 \def\current@section@level{document}%
4640 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4641 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4642 \cs_new_protected:Npn \skipomgroup {
4643   \ifcase\l_document_structure_section_level_int
4644   \or\stepcounter{part}
4645   \or\stepcounter{chapter}
4646   \or\stepcounter{section}
4647   \or\stepcounter{subsection}
4648   \or\stepcounter{subsubsection}
4649   \or\stepcounter{paragraph}
4650   \or\stepcounter{subparagraph}
4651   \fi
4652 }
```

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

`blindomgroup`

```
4653 \newcommand\at@begin@blindomgroup[1]{%
4654 \newenvironment{blindomgroup}
4655 {
4656   \int_incr:N\l_document_structure_section_level_int
4657   \at@begin@blindomgroup\l_document_structure_section_level_int
4658 }{}}
```

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

```
4659 \newcommand\omgroup@nonum[2]{
4660   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4661   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4662 }
```

(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.

```
4663 \newcommand\omgroup@num[2]{
```

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

```

4664 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4665   \@nameuse{#1}{#2}
4666 }{
4667   \cs_if_exist:NTF\rdfmata@sectioning{
4668     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4669   }{
4670     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4671   }
4672 }
4673 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4674 }

```

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

omgroup

```

4675 \keys_define:nn { document-structure / omgroup }{
4676   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4677   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4678   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4679   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4680   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4681   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4682   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4683   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4684   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4685   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4686 }
4687 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4688   \str_clear:N \l__document_structure_omgroup_id_str
4689   \str_clear:N \l__document_structure_omgroup_date_str
4690   \clist_clear:N \l__document_structure_omgroup_creators_clist
4691   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4692   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4693   \tl_clear:N \l__document_structure_omgroup_type_tl
4694   \tl_clear:N \l__document_structure_omgroup_short_tl
4695   \tl_clear:N \l__document_structure_omgroup_display_tl
4696   \tl_clear:N \l__document_structure_omgroup_intro_tl
4697   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4698   \keys_set:nn { document-structure / omgroup } { #1 }
4699 }

```

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.

```

4700 \newif\if@mainmatter\@mainmattertrue
4701 \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.

```

4702 \keys_define:nn { document-structure / sectioning }{
4703   name .str_set_x:N = \l__document_structure_sect_name_str ,
4704   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4705   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4706   num .bool_set:N = \l__document_structure_sect_num_bool ,
4707 }

```

```

4708 \cs_new_protected:Nn \__document_structure_sect_args:n {
4709   \str_clear:N \l__document_structure_sect_name_str
4710   \str_clear:N \l__document_structure_sect_ref_str
4711   \bool_set_false:N \l__document_structure_sect_clear_bool
4712   \bool_set_false:N \l__document_structure_sect_num_bool
4713   \keys_set:nn { document-structure / sectioning } { #1 }
4714 }
4715 \newcommand\omdoc@sectioning[3][]{
4716   \__document_structure_sect_args:n {#1}
4717   \let\omdoc@sect@name\l__document_structure_sect_name_str
4718   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4719   \if@mainmatter% numbering not overridden by frontmatter, etc.
4720     \bool_if:NTF \l__document_structure_sect_num_bool {
4721       \omgroup@num{#2}{#3}
4722     }{
4723       \omgroup@nonum{#2}{#3}
4724     }
4725     \def\current@section@level{\omdoc@sect@name}
4726   \else
4727     \omgroup@nonum{#2}{#3}
4728   \fi
4729 }% 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.

```

4730 \newcommand\omgroup@redefine@addtocontents[1]{%
4731   %\edef\__document_structureimport{#1}%
4732   %\@for\@I:=\__document_structureimport\do{%
4733     %\edef\@path{\csname module@\@I @path\endcsname}%
4734     %\@ifundefined{tf@toc}\relax%
4735     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4736   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4737   %\def\addcontentsline##1##2##3{%
4738     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4739   %\else% hyperref.sty not loaded
4740   %\def\addcontentsline##1##2##3{%
4741     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4742   %\fi
4743 }% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of `omgroups` in the `\omgroup@level` counter.

```

4744 \int_new:N \l_document_structure_omgroup_level_int
4745 \newenvironment{omgroup}[2][]{% keys, title
4746 {
4747   \__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.

```

4748 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4749   \omgroup@redefine@addtocontents{
4750     %\@ifundefined{module@id}\used@modules%
4751     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```



```

4752     }
4753 }

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

4754 \int_incr:N \l_document_structure_omgroup_level_int
4755 \int_incr:N \l_document_structure_section_level_int
4756 \ifcase\l_document_structure_section_level_int
4757   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4758   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4759   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4760   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4761   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4762   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4763   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4764 \fi
4765 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4766 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4767 }% for customization
4768 {}

```

and finally, we localize the sections

```

4769 \newcommand\omdoc@part@kw{Part}
4770 \newcommand\omdoc@chapter@kw{Chapter}
4771 \newcommand\omdoc@section@kw{Section}
4772 \newcommand\omdoc@subsection@kw{Subsection}
4773 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4774 \newcommand\omdoc@paragraph@kw{paragraph}
4775 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

38.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `omdoc` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

4776 \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).

```

4777 \cs_if_exist:NTF\frontmatter{
4778   \let\__document_structure_orig_frontmatter\frontmatter
4779   \let\frontmatter\relax
4780 }{
4781   \tl_set:Nn\__document_structure_orig_frontmatter{
4782     \clearpage
4783     \@mainmatterfalse
4784     \pagenumbering{roman}
4785   }
4786 }
4787 \cs_if_exist:NTF\backmatter{

```

```

4788 \let\__document_structure_orig_backmatter\backmatter
4789 \let\backmatter\relax
4790 }{
4791 \tl_set:Nn\__document_structure_orig_backmatter{
4792 \clearpage
4793 \@mainmatterfalse
4794 \pagenumbering{roman}
4795 }
4796 }

```

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.

```

4797 \newenvironment{frontmatter}{
4798 \__document_structure_orig_frontmatter
4799 }{
4800 \cs_if_exist:NTF\mainmatter{
4801 \mainmatter
4802 }{
4803 \clearpage
4804 \@mainmattertrue
4805 \pagenumbering{arabic}
4806 }
4807 }

```

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

```

4808 \newenvironment{backmatter}{
4809 \__document_structure_orig_backmatter
4810 }{
4811 \cs_if_exist:NTF\mainmatter{
4812 \mainmatter
4813 }{
4814 \clearpage
4815 \@mainmattertrue
4816 \pagenumbering{arabic}
4817 }
4818 }

```

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

```

4819 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4820 \def \c__document_structure_document_str{document}
4821 \newcommand\afterprematurestop{}
4822 \def\prematurestop@endomgroup{
4823 \unless\ifx\@currentvir\c__document_structure_document_str
4824 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4825 \expandafter\prematurestop@endomgroup
4826 \fi
4827 }
4828 \providecommand\prematurestop{

```

```

4829 \message{Stopping~sTeX~processing~prematurely}
4830 \prematurestop@endomgroup
4831 \afterprematurestop
4832 \end{document}
4833 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

4834 \RequirePackage{etoolbox}
4835 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4836 \newrobustcmd\useSGvar[1]{%
4837   \@ifundefined{sTeX@Gvar@#1}
4838   {\PackageError{omdoc}
4839    {The sTeX Global variable #1 is undefined}
4840    {set it with \protect\setSGvar}}
4841   \@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.

```

4842 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4843   \@ifundefined{sTeX@Gvar@#1}
4844   {\PackageError{omdoc}
4845    {The sTeX Global variable #1 is undefined}
4846    {set it with \protect\setSGvar}}
4847   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 39

MiKoSlides – Implementation

39.1 Class and Package Options

We define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `mikoslides` 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.

```
4848 \*cls)
4849 \@@=mikoslides)
4850 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4851 \RequirePackage{l3keys2e,expl-keystr-compatible}
4852
4853 \keys_define:nn{mikoslides / cls}{
4854   class .code:n = {
4855     \PassOptionsToClass{\CurrentOption}{omdoc}
4856     \str_if_eq:nnT{#1}{book}{
4857       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4858     }
4859     \str_if_eq:nnT{#1}{report}{
4860       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4861     }
4862   },
4863   notes .bool_set:N = \c__mikoslides_notes_bool ,
4864   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4865   unknown .code:n = {
4866     \PassOptionsToClass{\CurrentOption}{omdoc}
4867     \PassOptionsToClass{\CurrentOption}{beamer}
4868     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4869   }
4870 }
4871 \ProcessKeysOptions{ mikoslides / cls }
4872 \bool_if:NTF \c__mikoslides_notes_bool {
4873   \PassOptionsToPackage{notes=true}{mikoslides}
4874 }{
4875   \PassOptionsToPackage{notes=false}{mikoslides}
4876 }
4877 \</cls)
```

now we do the same for the mikoslides package.

```

4878 <*package>
4879 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4880 \RequirePackage{l3keys2e,expl-keystr-compat}
4881
4882 \keys_define:nn{mikoslides / pkg}{
4883   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4884   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4885   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4886   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4887   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4888   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4889   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4890   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4891   unknown      .code:n      = {
4892     \PassOptionsToClass{\CurrentOption}{stex}
4893     \PassOptionsToClass{\CurrentOption}{tikzinput}
4894   }
4895 }
4896 \ProcessKeysOptions{ mikoslides / pkg }
4897 \newif\ifnotes
4898 \bool_if:NTF \c__mikoslides_notes_bool {
4899   \notesttrue
4900 }{
4901   \notesfalse
4902 }
4903

```

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

```

4904 \str_if_empty:NTF \c__mikoslides_topsect_str {
4905   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4906 }{
4907   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4908 }
4909 </package>

```

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

```

4910 <*cls>
4911 \bool_if:NTF \c__mikoslides_notes_bool {
4912   \LoadClass{omdoc}
4913 }{
4914   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4915   \newcounter{Item}
4916   \newcounter{paragraph}
4917   \newcounter{subparagraph}
4918   \newcounter{Hfootnote}
4919   \RequirePackage{omdoc}
4920 }

```

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

```

4921 \RequirePackage{mikoslides}
4922 </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 `mikoslides.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.

```

4923 \*package>
4924 \bool_if:NT \c__mikoslides_notes_bool {
4925   \RequirePackage{a4wide}
4926   \RequirePackage{marginnote}
4927   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4928   \RequirePackage{mdframed}
4929   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4930   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4931 }
4932 \RequirePackage{stex-compatibility}
4933 \RequirePackage{stex-tikzinput}
4934 \RequirePackage{etoolbox}
4935 \RequirePackage{amssymb}
4936 \RequirePackage{amsmath}
4937 \RequirePackage{comment}
4938 \RequirePackage{textcomp}
4939 \RequirePackage{url}
4940 \RequirePackage{graphicx}
4941 \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

```

4942 \bool_if:NT \c__mikoslides_notes_bool {
4943   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4944 }

```

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

```

4945 \newcounter{slide}
4946 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4947 \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.

```

4948 \bool_if:NTF \c__mikoslides_notes_bool {
4949   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4950 }{
4951   \excludecomment{note}
4952 }

```

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.

```

4953 \bool_if:NT \c__mikoslides_notes_bool {
4954   \newlength{\slideframewidth}
4955   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

4956 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4957   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4958     \bool_set_true:N #1
4959   }{
4960     \bool_set_false:N #1
4961   }
4962 }
4963 \keys_define:nn{mikoslides / frame}{
4964   label .str_set_x:N = \l__mikoslides_frame_label_str,
4965   allowframebreaks .code:n = {
4966     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4967   },
4968   allowdisplaybreaks .code:n = {
4969     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4970   },
4971   fragile .code:n = {
4972     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4973   },
4974   shrink .code:n = {
4975     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4976   },
4977   squeeze .code:n = {
4978     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4979   },
4980   t .code:n = {
4981     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4982   },
4983 }
4984 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4985   \str_clear:N \l__mikoslides_frame_label_str
4986   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4987   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4988   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4989   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4990   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4991   \bool_set_true:N \l__mikoslides_frame_t_bool
4992   \keys_set:nn { mikoslides / frame }{ #1 }
4993 }

```

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

```

4994 \renewenvironment{frame}[1][]{
4995   \__mikoslides_frame_args:n{#1}
4996   \sffamily
4997   \stepcounter{slide}
4998   \def\@currentlabel{\theslide}
4999   \str_if_empty:NF \l__mikoslides_frame_label_str {
5000     \label{\l__mikoslides_frame_label_str}

```

5001 }

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

5002 \def\itemize@level{outer}
5003 \def\itemize@outer{outer}
5004 \def\itemize@inner{inner}
5005 \renewcommand\newpage{\addtocounter{framenum}{1}}
5006 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5007 \renewenvironment{itemize}{
5008 \ifx\itemize@level\itemize@outer
5009 \def\itemize@label{\\$ \rhd\$}
5010 \fi
5011 \ifx\itemize@level\itemize@inner
5012 \def\itemize@label{\\$ \scriptstyle \rhd\$}
5013 \fi
5014 \begin{list}
5015 {\itemize@label}
5016 {\setlength{\labelsep}{.3em}
5017 \setlength{\labelwidth}{.5em}
5018 \setlength{\leftmargin}{1.5em}
5019 }
5020 \edef\itemize@level{\itemize@inner}
5021 }{
5022 \end{list}
5023 }

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

5024 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5025 }{
5026 \medskip\miko@slidelabel\end{mdframed}
5027 }

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

\frametitle

5028 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5029 }

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

EdN:21

\pause 21

5030 \bool_if:NT \c__mikoslides_notes_bool {
5031 \newcommand\pause{
5032 }

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

nomtext

5033 \bool_if:NTF \c__mikoslides_notes_bool {
5034 \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5035 }{
5036 \excludecomment{nomtext}
5037 }

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

nomgroup

```
5038 \bool_if:NTF \c__mikoslides_notes_bool {
5039   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5040 }{
5041   \excludecomment{nomgroup}
5042 }
```

ndefinition

```
5043 \bool_if:NTF \c__mikoslides_notes_bool {
5044   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5045 }{
5046   \excludecomment{ndefinition}
5047 }
```

nassertion

```
5048 \bool_if:NTF \c__mikoslides_notes_bool {
5049   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5050 }{
5051   \excludecomment{nassertion}
5052 }
```

nsproof

```
5053 \bool_if:NTF \c__mikoslides_notes_bool {
5054   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5055 }{
5056   \excludecomment{nproof}
5057 }
```

nexample

```
5058 \bool_if:NTF \c__mikoslides_notes_bool {
5059   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5060 }{
5061   \excludecomment{nexample}
5062 }
```

\inputref@*skip We customize the hooks for in \inputref.

```
5063 \def\inputref@preskip{\smallskip}
5064 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5065 \let\orig@inputref\inputref
5066 \def\inputref{\@ifstar\ninputref\orig@inputref}
5067 \newcommand\ninputref[2] [] {
5068   \bool_if:NT \c__mikoslides_notes_bool {
5069     \orig@inputref[#1]{#2}
5070   }
5071 }
```

(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>}`.

```

5072 \newlength{\slidelogoheight}
5073
5074 \bool_if:NTF \c__mikoslides_notes_bool {
5075   \setlength{\slidelogoheight}{.4cm}
5076 }{
5077   \setlength{\slidelogoheight}{1cm}
5078 }
5079 \newsavebox{\slidelogo}
5080 \sbox{\slidelogo}{\TeX}
5081 \newrobustcmd{\setslidelogo}[1]{
5082   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5083 }
```

(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.

```

5084 \def\source{Michael Kohlhase}% customize locally
5085 \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.

```

5086 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5087 \newsavebox{\cclogo}
5088 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5089 \newif\ifcchref\cchreffalse
5090 \AtBeginDocument{
5091   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5092 }
5093 \def\licensing{
5094   \ifcchref
5095     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5096   \else
5097     {\usebox{\cclogo}}
5098   \fi
5099 }
5100 \newrobustcmd{\setlicensing}[2][]{
5101   \def@url{#1}
5102   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5103   \ifx@url@empty
5104     \def\licensing{{\usebox{\cclogo}}}
5105   \else
5106     \def\licensing{
```

```

5107     \ifcchref
5108     \href{#1}{\usebox{\cclogo}}
5109     \else
5110     {\usebox{\cclogo}}
5111     \fi
5112   }
5113 \fi
5114 }

```

(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

```

5115 \newrobustcmd\miko@slidelabel{
5116   \vbox to \slidelogoheight{
5117     \vss\hbox to \slidewidth
5118     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5119   }
5120 }

```

(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.

```

5121 \def\Gin@mhrepos{}
5122 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5123 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5124 \newrobustcmd\frameimage[2][{}{
5125   \stepcounter{slide}
5126   \bool_if:NT \c__mikoslides_frameimages_bool {
5127     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5128     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5129     \begin{center}
5130       \bool_if:NTF \c__mikoslides_fiboxed_bool {
5131         \fbox{
5132           \ifx\Gin@ewidth\@empty
5133             \ifx\Gin@mhrepos\@empty
5134               \mhgraphics[width=\slidewidth,#1]{#2}
5135             \else
5136               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5137             \fi
5138           \else% \Gin@ewidth empty
5139             \ifx\Gin@mhrepos\@empty
5140               \mhgraphics[#1]{#2}
5141             \else
5142               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5143             \fi
5144           \fi% \Gin@ewidth empty
5145         }
5146       }{
5147         \ifx\Gin@ewidth\@empty

```

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

```

5148         \ifx\Gin@mhrepos\@empty
5149             \mhgraphics[width=\slidewidth,#1]{#2}
5150         \else
5151             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5152         \fi
5153         \ifx\Gin@mhrepos\@empty
5154             \mhgraphics[#1]{#2}
5155         \else
5156             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5157         \fi
5158     \fi% Gin@ewidth empty
5159 }
5160 \end{center}
5161 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5162 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5163 }
5164 } % 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.

```

5165 \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

```

5166 \AddToHook{begindocument}{
5167     \definecolor{green}{rgb}{0,.5,0}
5168     \definecolor{purple}{cmyk}{.3,1,0,.17}
5169 }

```

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`.

```

5170 % \def\STpresent#1{\textcolor{blue}{#1}}
5171 \def\defemph#1{\textcolor{magenta}{#1}}
5172 \def\symrefemph#1{\textcolor{cyan}{#1}}
5173 \def\compemph#1{\textcolor{blue}{#1}}
5174 \def\titleemph#1{\textcolor{blue}{#1}}
5175 \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.

```

5176 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5177 \def\smalltextwarning{
5178     \pgfuseimage{miko@small@dbend}
5179     \xspace
5180 }
5181 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5182 \newrobustcmd\textwarning{
5183   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5184   \xspace
5185 }
5186 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5187 \newrobustcmd\bigtextwarning{
5188   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5189   \xspace
5190 }

```

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

```

5191 \newrobustcmd\putgraphicsat[3]{
5192   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5193 }
5194 \newrobustcmd\putat[2]{
5195   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5196 }

```

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`.

```

5197 \bool_if:NT \c__mikoslides_sectocframes_bool {
5198   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5199     \newcounter{chapter}\counterwithin*{section}{chapter}
5200   }{
5201     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5202       \newcounter{chapter}\counterwithin*{section}{chapter}
5203     }
5204   }
5205 }

```

`\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
5206 \def\part@prefix{}
5207 \@ifpackageloaded{omdoc}{}{
5208   \str_case:VnF \__mikoslidestopsect {
5209     {part}{
5210       \int_set:Nn \l_document_structure_section_level_int {0}
5211       \def\thesection{\arabic{chapter}.\arabic{section}}
5212       \def\part@prefix{\arabic{chapter}.}
5213     }
5214     {chapter}{
5215       \int_set:Nn \l_document_structure_section_level_int {1}
5216       \def\thesection{\arabic{chapter}.\arabic{section}}
5217       \def\part@prefix{\arabic{chapter}.}
5218     }
5219   }{
5220     \int_set:Nn \l_document_structure_section_level_int {2}
5221     \def\part@prefix{}

```

```

5222 }
5223 }
5224
5225 \bool_if:NF \c__mikoslides_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

```

5226 \renewenvironment{omgroup}[2][]{
5227   \__document_structure_omgroup_args:n { #1 }
5228   \int_incr:N \l_document_structure_omgroup_level_int
5229   \int_incr:N \l_document_structure_section_level_int
5230   \bool_if:NT \c__mikoslides_sectocframes_bool {
5231     \stepcounter{slide}
5232     \begin{frame}[noframenumbering]
5233     \vfill\Large\centering
5234     \red{
5235       \ifcase\l_document_structure_section_level_int\or
5236         \stepcounter{part}
5237         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5238         \def\currentsectionlevel{\omdoc@part@kw}
5239       \or
5240         \stepcounter{chapter}
5241         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5242         \def\currentsectionlevel{\omdoc@chapter@kw}
5243       \or
5244         \stepcounter{section}
5245         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5246         \def\currentsectionlevel{\omdoc@section@kw}
5247       \or
5248         \stepcounter{subsection}
5249         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5250         \def\currentsectionlevel{\omdoc@subsection@kw}
5251       \or
5252         \stepcounter{subsubsection}
5253         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5254         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5255       \or
5256         \stepcounter{paragraph}
5257         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5258         \def\currentsectionlevel{\omdoc@paragraph@kw}
5259       \else
5260         \def\__mikoslideslabel{}
5261         \def\currentsectionlevel{\omdoc@paragraph@kw}
5262       \fi% end ifcase
5263       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5264       \quad #2%
5265     }%
5266     \vfill%
5267     \end{frame}%
5268   }
5269   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5270 }{}
5271 }

```

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

```

5272 \def\inserttheorembodyfont{\normalfont}
5273 %\bool_if:NF \c__mikoslides_notes_bool {
5274 % \defbeamertemplate{theorem begin}{miko}
5275 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5276 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5277 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5278 % \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5279 % \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.

```

5280 % \expandafter\def\csname Parent2\endcsname{}
5281 %}
5282
5283 \AddToHook{begindocument}{% this does not work for some reasons
5284 \setbeamertemplate{theorems}[ams style]
5285 }
5286 \bool_if:NT \c__mikoslides_notes_bool {
5287 \renewenvironment{columns}[1][{}]{%
5288 \par\noindent%
5289 \begin{minipage}%
5290 \slidewidth\centering\leavevmode%
5291 }{%
5292 \end{minipage}\par\noindent%
5293 }%
5294 \newsavebox\columnbox%
5295 \renewenvironment<>{column}[2][{}]{%
5296 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5297 }{%
5298 \end{minipage}\end{lrbox}\usebox\columnbox%
5299 }%
5300 }
5301 \bool_if:NTF \c__mikoslides_noproblems_bool {
5302 \newenvironment{problems}{}{}
5303 }{
5304 \excludecomment{problems}
5305 }

```

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.

```

5306 \gdef\printexcursions{}
5307 \newcommand\excursionref[2]{% label, text
5308 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5309     \begin{sparagraph}[title=Excursion]
5310     #2 \sref[fallback=the appendix]{#1}.
5311     \end{sparagraph}
5312   }
5313 }
5314 \newcommand\activate@excursion[2][]{
5315   \gappto\printexcursions{\inputref[#1]{#2}}
5316 }
5317 \newcommand\excursion[4][]{% repos, label, path, text
5318   \bool_if:NT \c__mikoslides_notes_bool {
5319     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5320   }
5321 }

```

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

\excursiongroup

```

5322 \keys_define:nn{mikoslides / excursiongroup }{
5323   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
5324   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
5325   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5326 }
5327 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5328   \tl_clear:N \l__mikoslides_excursion_intro_tl
5329   \str_clear:N \l__mikoslides_excursion_id_str
5330   \str_clear:N \l__mikoslides_excursion_mhrepos_str
5331   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5332 }
5333 \newcommand\excursiongroup[1][]{
5334   \__mikoslides_excursion_args:n{ #1 }
5335   \ifdefempty\printexcursions{}% only if there are excursions
5336   {\begin{note}
5337     \begin{omgroup}[#1]{Excursions}%
5338     \ifdefempty\l__mikoslides_excursion_intro_tl{{
5339       \inputref[\l__mikoslides_excursion_mhrepos_str]{
5340         \l__mikoslides_excursion_intro_tl
5341       }
5342     }
5343     \printexcursions%
5344     \end{omgroup}
5345   }\end{note}}
5346 }
5347 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5348 \end{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.

```
5349 <*package>
5350 <@@=problems>
5351 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5352 \RequirePackage{l3keys2e,expl-keystr-compat}
5353
5354 \keys_define:nn { problem / pkg }{
5355   notes      .default:n    = { true },
5356   notes      .bool_set:N   = \c__problems_notes_bool,
5357   gnotes     .default:n    = { true },
5358   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5359   hints      .default:n    = { true },
5360   hints      .bool_set:N   = \c__problems_hints_bool,
5361   solutions  .default:n    = { true },
5362   solutions  .bool_set:N   = \c__problems_solutions_bool,
5363   pts        .default:n    = { true },
5364   pts        .bool_set:N   = \c__problems_pts_bool,
5365   min        .default:n    = { true },
5366   min        .bool_set:N   = \c__problems_min_bool,
5367   boxed      .default:n    = { true },
5368   boxed      .bool_set:N   = \c__problems_boxed_bool,
5369   unknown    .code:n       = {}
5370 }
5371 \def\solutionstrue{
5372   \bool_set_true:N \c__problems_solutions_bool
5373 }
5374 \def\solutionsfalse{
5375   \bool_set_false:N \c__problems_solutions_bool
5376 }
5377
5378 \ProcessKeysOptions{ problem / pkg }
```

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

```

5379 \RequirePackage{stex-compatibility}
5380 \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.

```

5381 \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.

```

5382 \def\prob@problem@kw{Problem}
5383 \def\prob@solution@kw{Solution}
5384 \def\prob@hint@kw{Hint}
5385 \def\prob@note@kw{Note}
5386 \def\prob@gnote@kw{Grading}
5387 \def\prob@pt@kw{pt}
5388 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5389 \@ifpackageloaded{babel}{
5390   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5391   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5392     \input{problem-ngerman.ldf}
5393   }
5394   \clist_if_in:NnT \l_tmpa_clist {finnish}{
5395     \input{problem-finnish.ldf}
5396   }
5397   \clist_if_in:NnT \l_tmpa_clist {french}{
5398     \input{problem-french.ldf}
5399   }
5400   \clist_if_in:NnT \l_tmpa_clist {russian}{
5401     \input{problem-russian.ldf}
5402   }
5403 }{}

```

40.2 Problems and Solutions

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

```

5404 \keys_define:nn{ problem / problem }{
5405   id      .str_set:x:N = \l__problems_prob_id_str,
5406   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5407   min     .tl_set:N    = \l__problems_prob_min_tl,
5408   title   .tl_set:N    = \l__problems_prob_title_tl,
5409   refnum  .int_set:N   = \l__problems_prob_refnum_int
5410 }
5411 \cs_new_protected:Nn \__problems_prob_args:n {
5412   \str_clear:N \l__problems_prob_id_str
5413   \tl_clear:N \l__problems_prob_pts_tl
5414   \tl_clear:N \l__problems_prob_min_tl
5415   \tl_clear:N \l__problems_prob_title_tl

```

```

5416 \int_zero_new:N \l__problems_prob_refnum_int
5417 \keys_set:nn { problem / problem }{ #1 }
5418 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5419   \let\l__problems_inclprob_refnum_int\undefined
5420 }
5421 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5422 \newcounter{problem}
5423 \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.

```

5424 \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

```

5425 \newcommand\prob@number{
5426   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5427     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5428   }{
5429     \int_if_exist:NTF \l__problems_prob_refnum_int {
5430       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5431     }{
5432       \prob@label\theproblem
5433     }
5434   }
5435 }

```

(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.

```

5436 \newcommand\prob@title[3]{%
5437   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5438     #2 \l__problems_inclprob_title_tl #3
5439   }{
5440     \tl_if_exist:NTF \l__problems_prob_title_tl {
5441       #2 \l__problems_prob_title_tl #3
5442     }{
5443       #1
5444     }
5445   }
5446 }

```

(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.

```
5447 \def\prob@heading{
5448   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5449   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
5450 }
```

(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.

`problem`

```
5451 \newenvironment{problem}[1][1]{
5452   \_problems_prob_args:n{#1}%\sref@target%
5453   \@in@omtexttrue% we are in a statement (for inline definitions)
5454   \stepcounter{problem}\record@problem
5455   \def\current@section@level{\prob@problem@kw}
5456   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5457 }%
5458 {\smallskip}
5459 \bool_if:NT \c__problems_boxed_bool {
5460   \surroundwithmdframed{problem}
5461 }
```

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

```
5462 \def\record@problem{
5463   \protected@write\@auxout{}
5464   {
5465     \string\@problem{\prob@number}
5466     {
5467       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5468         \l__problems_inclprob_pts_tl
5469       }{
5470         \l__problems_prob_pts_tl
5471       }
5472     }%
5473     {
5474       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5475         \l__problems_inclprob_min_tl
5476       }{
5477         \l__problems_prob_min_tl
5478       }
5479     }
5480   }
5481 }
```

(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).

```
5482 \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.

```

5483 \keys_define:nn { problem / solution }{
5484   id          .str_set_x:N = \l__problems_solution_id_str ,
5485   for         .tl_set:N    = \l__problems_solution_for_tl ,
5486   height      .dim_set:N   = \l__problems_solution_height_dim ,
5487   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5488   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5489   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5490 }
5491 \cs_new_protected:Nn \__problems_solution_args:n {
5492   \str_clear:N \l__problems_solution_id_str
5493   \tl_clear:N \l__problems_solution_for_tl
5494   \tl_clear:N \l__problems_solution_srccite_tl
5495   \clist_clear:N \l__problems_solution_creators_clist
5496   \clist_clear:N \l__problems_solution_contributors_clist
5497   \dim_zero:N \l__problems_solution_height_dim
5498   \keys_set:nn { problem / solution }{ #1 }
5499 }

```

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

```

5500 \newcommand\@startsolution[1][ ]{
5501   \__problems_solution_args:n { #1 }
5502   \@in@omtexttrue% we are in a statement.
5503   \bool_if:NF \c__problems_boxed_bool { \hrule }
5504   \smallskip\noindent
5505   {\textbf\prob@solution@kw : \enspace}
5506   \begin{small}
5507   \def\current@section@level{\prob@solution@kw}
5508   \ignorespacesandpars
5509 }

```

\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.

```

5510 \newcommand\startsolutions{
5511   \specialcomment{solution}{\@startsolution}{
5512     \bool_if:NF \c__problems_boxed_bool {
5513       \hrule\medskip
5514     }
5515     \end{small}%
5516   }
5517   \bool_if:NT \c__problems_boxed_bool {
5518     \surroundwithmdframed{solution}
5519   }
5520 }

```

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

\stopsolutions

```

5521 \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.

```

5522 \bool_if:NTF \c__problems_solutions_bool {
5523   \startsolutions
5524 }{
5525   \stopsolutions
5526 }

```

exnote

```

5527 \bool_if:NTF \c__problems_notes_bool {
5528   \newenvironment{exnote}[1][]{
5529     \par\smallskip\hrule\smallskip
5530     \noindent\textbf{\prob@note@kw : }\small
5531   }{
5532     \smallskip\hrule
5533   }
5534 }{
5535   \excludecomment{exnote}
5536 }

```

hint

```

5537 \bool_if:NTF \c__problems_notes_bool {
5538   \newenvironment{hint}[1][]{
5539     \par\smallskip\hrule\smallskip
5540     \noindent\textbf{\prob@hint@kw :~ }\small
5541   }{
5542     \smallskip\hrule
5543   }
5544   \newenvironment{exhint}[1][]{
5545     \par\smallskip\hrule\smallskip
5546     \noindent\textbf{\prob@hint@kw :~ }\small
5547   }{
5548     \smallskip\hrule
5549   }
5550 }{
5551   \excludecomment{hint}
5552   \excludecomment{exhint}
5553 }

```

gnote

```

5554 \bool_if:NTF \c__problems_notes_bool {
5555   \newenvironment{gnote}[1][]{
5556     \par\smallskip\hrule\smallskip
5557     \noindent\textbf{\prob@gnote@kw : }\small
5558   }{
5559     \smallskip\hrule
5560   }
5561 }{
5562   \excludecomment{gnote}
5563 }

```

40.3 Multiple Choice Blocks

```

5564 \newenvironment{mcb}{
5565   \begin{enumerate}
5566 }{
5567   \end{enumerate}
5568 }

```

we define the keys for the mcc macro

```

5569 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5570   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5571     \bool_set_true:N #1
5572   }{
5573     \bool_set_false:N #1
5574   }
5575 }
5576 \keys_define:nn { problem / mcc }{
5577   id          .str_set:x:N = \l__problems_mcc_id_str ,
5578   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5579   T           .default:n    = { true } ,
5580   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5581   F           .default:n    = { true } ,
5582   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5583   Ttext       .code:n       = {
5584     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5585   } ,
5586   Ftext       .code:n       = {
5587     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5588   }
5589 }
5590 \cs_new_protected:Nn \l__problems_mcc_args:n {
5591   \str_clear:N \l__problems_mcc_id_str
5592   \tl_clear:N \l__problems_mcc_feedback_tl
5593   \bool_set_true:N \l__problems_mcc_t_bool
5594   \bool_set_true:N \l__problems_mcc_f_bool
5595   \bool_set_true:N \l__problems_mcc_Ttext_bool
5596   \bool_set_false:N \l__problems_mcc_Ftext_bool
5597   \keys_set:nn { problem / mcc }{ #1 }
5598 }

```

\mcc

```

5599 \newcommand\mcc[2][] {
5600   \l__problems_mcc_args:n{ #1 }
5601   \item #2
5602   \bool_if:NT \c__problems_solutions_bool {
5603     \\\
5604     \bool_if:NT \l__problems_mcc_t_bool {
5605       % TODO!
5606       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5607     }
5608     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

5609      % TODO!
5610      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5611    }
5612    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5613      !
5614    }{
5615      \l__problems_mcc_feedback_tl
5616    }
5617  }
5618 } %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.

```

5619
5620 \keys_define:nn{ problem / inclproblem }{
5621   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5622   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5623   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5624   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5625   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5626   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5627 }
5628 \cs_new_protected:Nn \__problems_inclprob_args:n {
5629   % \str_clear:N \l__problems_prob_id_str
5630   \tl_clear:N \l__problems_inclprob_pts_tl
5631   \tl_clear:N \l__problems_inclprob_min_tl
5632   \tl_clear:N \l__problems_inclprob_title_tl
5633   \int_zero_new:N \l__problems_inclprob_refnum_int
5634   \str_clear:N \l__problems_inclprob_mhrepos_str
5635   \keys_set:nn { problem / inclproblem }{ #1 }
5636   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5637     \let\l__problems_inclprob_pts_tl\undefined
5638   }
5639   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5640     \let\l__problems_inclprob_min_tl\undefined
5641   }
5642   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5643     \let\l__problems_inclprob_title_tl\undefined
5644   }
5645   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5646     \let\l__problems_inclprob_refnum_int\undefined
5647   }
5648 }
5649
5650 \cs_new_protected:Nn \__problems_inclprob_clear: {
5651   % \str_clear:N \l__problems_prob_id_str
5652   \let\l__problems_inclprob_pts_tl\undefined
5653   \let\l__problems_inclprob_min_tl\undefined

```



```

5654 \let\l__problems_inclprob_title_tl\undefined
5655 \let\l__problems_inclprob_refnum_int\undefined
5656 \let\l__problems_inclprob_mhrepos_str\undefined
5657 }
5658
5659 \newcommand\includeproblem[2][]{
5660   \__problems_inclprob_args:n{ #1 }
5661   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5662     \input{#2}
5663   }{
5664     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5665       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5666     }
5667   }
5668   \__problems_inclprob_clear:
5669 }

```

(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.

```

5670 \AddToHook{enddocument}{
5671   \bool_if:NT \c__problems_pts_bool {
5672     \message{Total:~\arabic{pts}~points}
5673   }
5674   \bool_if:NT \c__problems_min_bool {
5675     \message{Total:~\arabic{min}~minutes}
5676   }
5677 }

```

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

```

5678 \def\pts#1{
5679   \bool_if:NT \c__problems_pts_bool {
5680     \marginpar{#1~\prob@pt@kw}
5681   }
5682 }
5683 \def\min#1{
5684   \bool_if:NT \c__problems_min_bool {
5685     \marginpar{#1~\prob@min@kw}
5686   }
5687 }

```

\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.

```

5688 \newcounter{pts}
5689 \def\show@pts{
5690   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5691     \bool_if:NT \c__problems_pts_bool {
5692       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5693       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5694     }
5695   }{
5696     \tl_if_exist:NT \l__problems_prob_pts_tl {
5697       \bool_if:NT \c__problems_pts_bool {
5698         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5699         \addtocounter{pts}{\l__problems_prob_pts_tl}
5700       }
5701     }
5702   }
5703 }

```

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

\show@min

```

5704 \newcounter{min}
5705 \def\show@min{
5706   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5707     \bool_if:NT \c__problems_min_bool {
5708       \marginpar{\l__problems_inclprob_pts_tl;min}
5709       \addtocounter{min}{\l__problems_inclprob_min_tl}
5710     }
5711   }{
5712     \tl_if_exist:NT \l__problems_prob_min_tl {
5713       \bool_if:NT \c__problems_min_bool {
5714         \marginpar{\l__problems_prob_min_tl;min}
5715         \addtocounter{min}{\l__problems_prob_min_tl}
5716       }
5717     }
5718   }
5719 }
5720 \</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.

```
5721 <@@=hwexam>
5722 <*cls>
5723 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5724 \RequirePackage{l3keys2e,expl-keystr-compatible}
5725 \DeclareOption*{
5726   \PassOptionsToClass{\CurrentOption}{omdoc}
5727   \PassOptionsToPackage{\CurrentOption}{stex}
5728   \PassOptionsToPackage{\CurrentOption}{hwexam}
5729   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5730 }
5731 \ProcessOptions
```

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

```
5732 \LoadClass{omdoc}
5733 \RequirePackage{stex}
5734 \RequirePackage{hwexam}
5735 \RequirePackage{tikzinput}
5736 \RequirePackage{graphicx}
5737 \RequirePackage{a4wide}
5738 \RequirePackage{amssymb}
5739 \RequirePackage{amstext}
5740 \RequirePackage{amsmath}
```

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

```

5741 \newcommand\assig@default@type{\hwexam@assignment@kw}
5742 \def\document@hwexamtype{\assig@default@type}
5743 <@@=document_structure>
5744 \keys_define:nn { document-structure / document }{
5745 id .str_set_x:N = \c_document_structure_document_id_str,
5746 hwexamtype .tl_set:N = \document@hwexamtype
5747 }
5748 <@@=hwexam>
5749 </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.

```
5750 \langle *package\rangle
5751 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5752 \RequirePackage{l3keys2e,expl-keystr-compat}
5753
5754 \newif\iftest\testfalse
5755 \DeclareOption{test}{\testtrue}
5756 \newif\ifmultiple\multiplefalse
5757 \DeclareOption{multiple}{\multipletrue}
5758 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5759 \ProcessOptions
```

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

```
5760 \RequirePackage{keyval}[1997/11/10]
5761 \RequirePackage{problem}
```

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

```
5762 \newcommand\hwexam@assignment@kw{Assignment}
5763 \newcommand\hwexam@given@kw{Given}
5764 \newcommand\hwexam@due@kw{Due}
5765 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5766 blank~for~extra~space}%
5767 \newcommand\correction@probs@kw{prob.}%
5768 \newcommand\correction@pts@kw{total}%
5769 \newcommand\correction@reached@kw{reached}%
5770 \newcommand\correction@sum@kw{Sum}%
5771 \newcommand\correction@grade@kw{grade}%
5772 \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

```

5773 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5774
5775 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5776 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5777   \input{hwexam-ngerman.ldf}
5778 }
5779 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5780   \input{hwexam-finnish.ldf}
5781 }
5782 \clist_if_in:NnT \l_tmpa_clist {french}{
5783   \input{hwexam-french.ldf}
5784 }
5785 \clist_if_in:NnT \l_tmpa_clist {russian}{
5786   \input{hwexam-russian.ldf}
5787 }

```

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.

```

5788 \newcounter{assignment}
5789 \numberproblemsin{assignment}
5790 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5791 \keys_define:nn { hwexam / assignment } {
5792   id .str_set:N = \l__hwexam_assign_id_str,
5793   number .int_set:N = \l__hwexam_assign_number_int,
5794   title .tl_set:N = \l__hwexam_assign_title_tl,
5795   type .tl_set:N = \l__hwexam_assign_type_tl,
5796   given .tl_set:N = \l__hwexam_assign_given_tl,
5797   due .tl_set:N = \l__hwexam_assign_due_tl,
5798   loadmodules .code:n = {
5799     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5800   }
5801 }
5802 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5803   \str_clear:N \l__hwexam_assign_id_str
5804   \int_set:Nn \l__hwexam_assign_number_int {-1}
5805   \tl_clear:N \l__hwexam_assign_title_tl
5806   \tl_clear:N \l__hwexam_assign_type_tl
5807   \tl_clear:N \l__hwexam_assign_given_tl
5808   \tl_clear:N \l__hwexam_assign_due_tl
5809   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5810   \keys_set:nn { hwexam / assignment }{ #1 }
5811 }

```

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.

```

5812 \newcommand\given@due[2]{
5813 \bool_lazy_all:nF {
5814 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5815 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5816 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5817 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5818 }{ #1 }
5819
5820 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5821 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5822 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5823 }
5824 }{
5825 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5826 }
5827
5828 \bool_lazy_or:nnF {
5829 \bool_lazy_and_p:nn {
5830 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5831 }{
5832 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5833 }
5834 }{
5835 \bool_lazy_and_p:nn {
5836 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5837 }{
5838 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5839 }
5840 }{ ,~ }
5841
5842 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5843 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5844 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5845 }
5846 }{
5847 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5848 }
5849
5850 \bool_lazy_all:nF {
5851 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5852 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5853 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5854 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5855 }{ #2 }
5856 }

```

`\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.

```

5857 \newcommand\assignment@title[3]{

```

```

5858 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5859 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5860 #1
5861 }{
5862 #2\l__hwexam_assign_title_tl#3
5863 }
5864 }{
5865 #2\l__hwexam_inclassassign_title_tl#3
5866 }
5867 }

```

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

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

```

5868 \newcommand\assignment@number{
5869 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5870 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5871 \int_use:N \l__hwexam_assign_number_int
5872 }
5873 }{
5874 \int_use:N \l__hwexam_inclassassign_number_int
5875 }
5876 }

```

(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.

```

5877 \newenvironment{assignment}[1][]{
5878 \__hwexam_assignment_args:n { #1 }
5879 %\sref@target
5880 \let\__hwexamnum\l__hwexam_assign_number_int
5881 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5882 \stepcounter{assignment}
5883 }{
5884 \setcounter{assignment}{\int_use:N\__hwexamnum}
5885 }
5886 \setcounter{problem}{0}
5887 \def\current@section@level{\document@hwexamtype}
5888 %\sref@label@id{\document@hwexamtype \thesection}
5889 \begin{@assignment}
5890 }{
5891 \end{@assignment}
5892 }

```

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

```

5893 \def\__hwexasstitle{
5894 \protect\document@hwexamtype~\arabic{assignment}
5895 \assignment@title{}\;{} \; -- \given@due{}\}
5896 }

```



```

5897 \ifmultiple
5898 \newenvironment{@assignment}{
5899 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5900 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5901 }{
5902 \begin{omgroup}{\__hwexasstitle}
5903 }
5904 }{
5905 \end{omgroup}
5906 }

```

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

```

5907 \else
5908 \newenvironment{@assignment}{
5909 \begin{center}\bf
5910 \Large\@title\strut\
5911 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5912 \large\given@due{--\;}{\;}{--}
5913 \end{center}
5914 }{}
5915 \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.

```

5916 \keys_define:nn { hwexam / inclassignment } {
5917 %id .str_set_x:N = \l__hwexam_assign_id_str,
5918 number .int_set:N = \l__hwexam_inclassign_number_int,
5919 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5920 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5921 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5922 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5923 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5924 }
5925 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5926 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5927 \tl_clear:N \l__hwexam_inclassign_title_tl
5928 \tl_clear:N \l__hwexam_inclassign_type_tl
5929 \tl_clear:N \l__hwexam_inclassign_given_tl
5930 \tl_clear:N \l__hwexam_inclassign_due_tl
5931 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5932 \keys_set:nn { hwexam / inclassignment }{ #1 }
5933 }
5934 \__hwexam_inclassignment_args:n {}
5935
5936 \newcommand\inputassignment[2][ ]{
5937 \__hwexam_inclassignment_args:n { #1 }
5938 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5939 \input{#2}
5940 }{
5941 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5942 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5943 }
5944 }
5945 \__hwexam_inclasssignment_args:n {}
5946 }
5947 \newcommand\includeassignment[2][ ]{
5948 \newpage
5949 \inputassignment[#1]{#2}
5950 }

```

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

42.4 Typesetting Exams

\quizheading

```

5951 \ExplSyntaxOff
5952 \newcommand\quizheading[1]{%
5953 \def\@tas{#1}%
5954 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5955 \ifx\@tas\empty\else%
5956 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5957 \fi%
5958 }
5959 \ExplSyntaxOn

```

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

\testheading

```

5960 \keys_define:nn { hwexam / testheading } {
5961 min .tl_set:N = \l__hwexam_testheading_min_tl,
5962 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5963 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5964 }
5965 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5966 \tl_clear:N \l__hwexam_testheading_min_tl
5967 \tl_clear:N \l__hwexam_testheading_duration_tl
5968 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5969 \keys_set:nn { hwexam / testheading }{ #1 }
5970 }
5971 \newenvironment{testheading}[1][ ]{
5972 \__hwexam_testheading_args:n{ #1 }
5973 \noindent\large{Name:~\hfill
5974 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5975 \begin{center}
5976 \Large\textbf{\@title}\[1ex]
5977 \large\@date\[3ex]
5978 \end{center}
5979 \textbf{You~have~
5980 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5981 {\l__hwexam_testheading_min_tl}~minutes
5982 }{
5983 {\l__hwexam_testheading_duration_tl}
5984 }~

```

```

5985 (sharp)~for~the~test
5986 };\
5987 Write~the~solutions~to~the~sheet.
5988 \par\noindent
5989 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5990 \advance\check@time by -\theassignment@totalmin
5991 The~estimated~time~for~solving~this~exam~is~
5992 {\theassignment@totalmin}~minutes,~
5993 leaving~you~{\the\check@time}~minutes~for~revising~
5994 your~exam.
5995
5996 \par\noindent
5997 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5998 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5999 You~can~reach~{\theassignment@totalpts}~points~if~you~
6000 solve~all~problems.~You~will~only~need~
6001 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
6002 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
6003 \vfill
6004 \begin{center}
6005 {
6006 \Large\em You~have~ample~time,~so~take~it~slow~
6007 and~avoid~rushing~to~mistakes!\}[2ex]
6008 Different~problems~test~different~skills~and~
6009 knowledge,~so~do~not~get~stuck~on~one~problem.
6010 }
6011 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
6012 \end{center}
6013 }{
6014 \newpage
6015 }

```

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

\testspace

```

6016 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6017 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6018 \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.

```

6019 <@=problems>
6020 \renewcommand\@problem[3]{
6021 \stepcounter{assignment@probs}
6022 \def\__problemspts{#2}

```

```

6023 \ifx\__problemspts\@empty\else
6024 \addtocounter{assignment@totalpts}{#2}
6025 \fi
6026 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6027 \xdef\correction@probs{\correction@probs & #1}%
6028 \xdef\correction@pts{\correction@pts & #2}
6029 \xdef\correction@reached{\correction@reached & }
6030 }
6031 \<@=hwexam>

```

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

\correction@table This macro generates the correction table

```

6032 \newcounter{assignment@probs}
6033 \newcounter{assignment@totalpts}
6034 \newcounter{assignment@totalmin}
6035 \def\correction@probs{\correction@probs@kw}%
6036 \def\correction@pts{\correction@pts@kw}%
6037 \def\correction@reached{\correction@reached@kw}%
6038 \def\after@correction@table{}%
6039 \stepcounter{assignment@probs}
6040 \newcommand\correction@table{
6041 \resizebox{\textwidth}{!}{%
6042 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6043 &\multicolumn{\theassignment@probs}{c|}||%|
6044 {\footnotesize\correction@forgrading@kw} &\\ \hline
6045 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6046 \correction@pts & \theassignment@totalpts & \\ \hline
6047 \correction@reached & & \[.7cm]\hline
6048 \end{tabular}}
6049 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6050 \end{package}

```

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

42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\wierfont=../assignments/wierfont
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\wierfont{\font\wierfont\char65}
\newcommand\denkerfont{\font\denkerfont\char65}
\newcommand\uhrfont{\font\uhrfont\char65}
\newcommand\warnschildfont{\font\warnschildfont\char 65}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\wierfont}

```