

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	122
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	156
33.1	Definitions	156
33.2	Assertions	159
33.3	Examples	161
33.4	Logical Paragraphs	163

34 The Implementation	166
34.1 Package Options	166
34.2 Proofs	166
34.3 Justifications	172
35 \TeX-Others Implementation	174
36 \TeX-Metatheory Implementation	175
37 Tikzinput Implementation	178
38 document-structure.sty Implementation	180
38.1 The OMDoc Class	180
38.2 Class Options	180
38.3 Beefing up the <code>document</code> environment	181
38.4 Implementation: OMDoc Package	181
38.5 Package Options	181
38.6 Document Structure	183
38.7 Front and Backmatter	186
38.8 Global Variables	188
39 MiKoSlides – Implementation	189
39.1 Class and Package Options	189
39.2 Notes and Slides	191
39.3 Header and Footer Lines	195
39.4 Frame Images	196
39.5 Colors and Highlighting	197
39.6 Sectioning	198
39.7 Excursions	200
40 The Implementation	202
40.1 Package Options	202
40.2 Problems and Solutions	203
40.3 Multiple Choice Blocks	208
40.4 Including Problems	209
40.5 Reporting Metadata	210
41 Implementation: The hwexam Class	212
41.1 Class Options	212
42 Implementation: The hwexam Package	214
42.1 Package Options	214
42.2 Assignments	215
42.3 Including Assignments	218
42.4 Typesetting Exams	219
42.5 Leftovers	221

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First sTeX Document

Having set everything up, we can write a first sTeX 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 sTeX *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}`), sTeX 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}`.

sTeX 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: $\rtimes a{\rplus c}{\rtimes de}$
\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}]{ $a$ }{ $b$ }[ \comp{and} ]{ $b$ }
```

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

.

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

Example 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:NTF</code>	\star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

10.1.2 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

Namespace 1:
file://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⟩}`

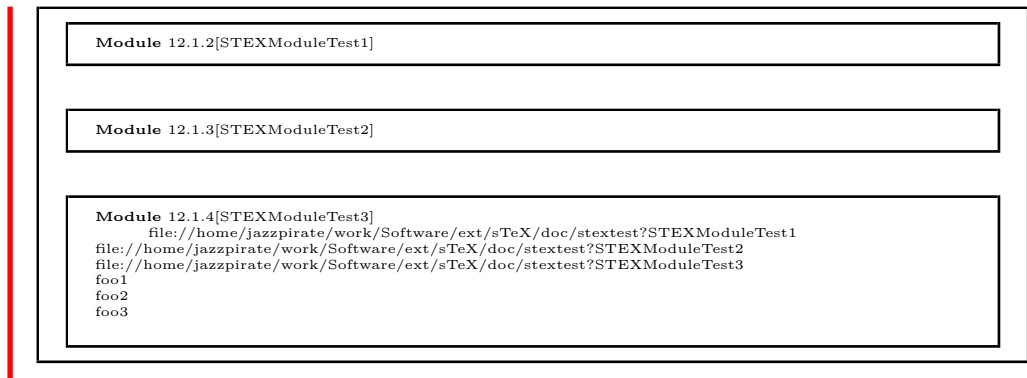
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\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>,
 All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collect>,
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?seqtype>,
<http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?module-type>,
<http://mathhub.info/sTeX?Metatheory?mathematical-structure>,
<http://mathhub.info/sTeX?Metatheory?dummyvar>,
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collect>,
<http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?module-type>,
<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

<code>\symdecl</code>	<code>\symdecl[$\langle args \rangle$]{$\langle macroname \rangle$}</code>
-----------------------	--

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to $\langle macroname \rangle$.
- **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. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\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. `\symdecl[args=bi]{forall}` allows for `\forall{x \in \mathbb{N}}{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 } }
 $\bar{foo} a\{b,c,d,e,f\}g$  and  $\bar{foo} a\{b,c\}g$  and  $\bar{foo} abc$ 
\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{#1 \comp+ #2}
\notation[prec=100]{ mult }{#1}{#1 \comp\cdot #2}
 $\plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{c}}}$ 
 $\displaystyle \plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{c}}}$ 
\withbrackets[] {  $\displaystyle \mult{a,\plus{\frac{ab}{c}}}$  }
\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

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

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

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}`).

`\sProofEndSymbol`

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:

EdN:8

Environment	configuration macro	value
<code>sproof</code>	<code>\spf@proof@kw</code>	Proof
<code>sketchproof</code>	<code>\spf@sketchproof@kw</code>	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 L^AT_EX `\@for...:=... \do{...}` macro; see Figure 2 for examples.

style	example	configuration macro
<code>long</code>	<code>0.8.1.5</code>	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code>
<code>angles</code>	<code>>>>5</code>	<code>\def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}</code>
<code>short</code>	<code>5</code>	<code>\def\pst@make@label@short#1#2{#2}</code>
<code>empty</code>		<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 S_TE_X 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 \LaTeX collection, 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).

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

21.1 Introduction

\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). 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 \LaTeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the \LaTeX collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document

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.barderv,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` [`\URL`]{`\label`}, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL [`\URL`] 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`{`\vname`}{`\text`} to set the global variable `\vname` to `\text` and `\useSGvar`{`\vname`} to reference it.

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

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> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 22.2.2). |
| <code>sectocframes</code> | • 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, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the [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 referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV
Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

25.2 Preliminaries

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

Package options:

```

26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
28   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_EX_ML, R_US_TE_X, p_DF_LA_TE_X).

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

```

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:nn`

```

281 <@@=stex_aftergroup>
282 \tl_new:N \l__stex_aftergroup_tl
283 \cs_new_protected:Nn \stex_do_aftergroup:n {
284   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
285     #1
286   }{
287     #1
288     \expandafter \tl_gset:Nn \expandafter \l__stex_aftergroup_tl \expandafter { \l__stex_aft
289     \aftergroup\__stex_aftergroup_do:
290   }
291 }
292 \cs_new_protected:Nn \__stex_aftergroup_do: {
293   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
294     \l__stex_aftergroup_tl
295     \tl_clear:N \l__stex_aftergroup_tl
296   }{
297     \l__stex_aftergroup_tl
298     \aftergroup\__stex_aftergroup_do:
299   }
300 }

```

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

```

301 </package>

```

Chapter 26

STEX -MathHub Implementation

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

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
317 \cs_new_protected:Nn \stex_path_from_string:Nn {
318   \str_set:Nx \l_tmpa_str { #2 }
319   \str_if_empty:NTF \l_tmpa_str {
320     \seq_clear:N #1
321   }{
322     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
323     \sys_if_platform_windows:T{
324       \seq_clear:N \l_tmpa_tl
325       \seq_map_inline:Nn #1 {
326         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
327         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
328       }
329     }
330   }
```

```

328     }
329     \seq_set_eq:NN #1 \l_tmpa_tl
330   }
331   \stex_path_canonicalize:N #1
332 }
333 }
334 \cs_generate_variant:Nn \stex_path_from_string:Nn
335 { 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
336 \cs_new_protected:Nn \stex_path_to_string:NN {
337   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
338 }
339
340 \cs_new:Nn \stex_path_to_string:N {
341   \seq_use:Nn #1 /
342 }

```

(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
343 \str_const:Nn \c__stex_path_dot_str {.}
344 \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.

```

345 \cs_new_protected:Nn \stex_path_canonicalize:N {
346   \seq_if_empty:NF #1 {
347     \seq_clear:N \l_tmpa_seq
348     \seq_get_left:NN #1 \l_tmpa_tl
349     \str_if_empty:NT \l_tmpa_tl {
350       \seq_put_right:Nn \l_tmpa_seq {}
351     }
352     \seq_map_inline:Nn #1 {
353       \str_set:Nn \l_tmpa_tl { ##1 }
354       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
355         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
356           \seq_if_empty:NTF \l_tmpa_seq {
357             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
358               \c__stex_path_up_str
359             }
360           }{
361             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
362             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
363               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
364                 \c__stex_path_up_str
365               }
366             }{
367               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
368             }

```

```

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

```

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

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

```

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

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(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`

```

398 \sys_if_platform_windows:TF{
399   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
400 }{
401   \stex_kpsewhich:n{-var-value~PWD}
402 }
403
404 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
405 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
406 \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

407 `<@@=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

408 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

410 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

411 `\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`.

```

412 \seq_gclear_new:N\g_stex_currentfile_seq
413 \AddToHook{file/before}{
414   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
415   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
416     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
417   }{
418     \stex_path_from_string:Nn\g_stex_currentfile_seq{
419       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
420     }
421   }
422   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
423   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
424 }
425 \AddToHook{file/after}{
426   \seq_if_empty:NF\g__stex_files_stack{
427     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
428   }
429   \seq_if_empty:NTF\g__stex_files_stack{
430     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
431   }{
432     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
433     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
434   }
435 }

```

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

26.4 MathHub Repositories

```

436 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
437 \str_if_empty:NTF\mathhub{
438   \stex_kpsewhich:n{-var-value~MATHHUB}
439   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
440
441   \str_if_empty:NTF\c_stex_mathhub_str{
442     \msg_warning:nn{stex}{warning/nomathhub}
443   }{
444     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
445     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
446   }
447 }{
448   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
449   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
450     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
451       \c_stex_pwd_str/\mathhub
452     }
453   }
454   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
455   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
456 }

```

(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
457 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
458   \str_set:Nx \l_tmpa_str { #1 }
459   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
460     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
461     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
462     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
463     \__stex_mathhub_find_manifest:N \l_tmpa_seq
464     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
465       \msg_error:nnxx{stex}{error/norepository}{#1}{
466         \stex_path_to_string:N \c_stex_mathhub_str
467       }
468     } {
469       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
470     }
471   }
472 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
473 \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`:

```

474 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
475   \seq_set_eq:NN \l_tmpa_seq #1
476   \bool_set_true:N \l_tmpa_bool
477   \bool_while_do:Nn \l_tmpa_bool {
478     \seq_if_empty:NTF \l_tmpa_seq {
479       \bool_set_false:N \l_tmpa_bool
480     }{
481       \file_if_exist:nTF{
482         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
483       }{
484         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
485         \bool_set_false:N \l_tmpa_bool
486       }{
487         \file_if_exist:nTF{
488           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
489         }{
490           \seq_put_right:Nn \l_tmpa_seq{META-INF}
491           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
492           \bool_set_false:N \l_tmpa_bool
493         }{
494           \file_if_exist:nTF{
495             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
496           }{
497             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
498             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
499             \bool_set_false:N \l_tmpa_bool
500           }{
501             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
502           }
503         }
504       }
505     }
506   }
507   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
508 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```
509 \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:

```

510 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
511   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
512   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
513   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
514     \str_set:Nn \l_tmpa_str {##1}
515     \exp_args:NNoo \seq_set_split:Nnn
516       \l_tmpb_seq \c_colon_str \l_tmpa_str
517     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

518 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
519 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
520 }
521 \exp_args:No \str_case:nnTF \l_tmpa_tl {
522 {id} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { id } \l_tmpb_tl
525 }
526 {narration-base} {
527 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528 { narr } \l_tmpb_tl
529 }
530 {url-base} {
531 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532 { docurl } \l_tmpb_tl
533 }
534 {source-base} {
535 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
536 { ns } \l_tmpb_tl
537 }
538 {ns} {
539 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
540 { ns } \l_tmpb_tl
541 }
542 {dependencies} {
543 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
544 { deps } \l_tmpb_tl
545 }
546 }{}{}
547 }{}
548 }
549 \ior_close:N \c__stex_mathhub_manifest_ior
550 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

551 \cs_new_protected:Nn \stex_set_current_repository:n {
552 \stex_require_repository:n { #1 }
553 \prop_set_eq:Nc \l_stex_current_repository_prop {
554 c_stex_mathhub_#1_manifest_prop
555 }
556 }

```

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

`\stex_require_repository:n`

```

557 \cs_new_protected:Nn \stex_require_repository:n {
558 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
559 \stex_debug:nn{mathhub}{Opening~archive:~#1}
560 \__stex_mathhub_do_manifest:n { #1 }
561 \exp_args:Nx \stex_add_to_sms:n {
562 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
563 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
564 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

565     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
566     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
567   }
568 }
569 }
570 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

571 %\prop_new:N \l_stex_current_repository_prop
572
573 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
574 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
575   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
576 } {
577   \__stex_mathhub_parse_manifest:n { main }
578   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
579   \l_tmpa_str
580   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
581   \c_stex_mathhub_main_manifest_prop
582   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
583   \stex_debug:nn{mathhub}{Current~repository:~
584   \prop_item:Nn \l_stex_current_repository_prop {id}
585   }
586 }

```

(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.

```

587 \cs_new_protected:Nn \stex_in_repository:nn {
588   \str_set:Nx \l_tmpa_str { #1 }
589   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
590   \str_if_empty:NTF \l_tmpa_str {
591     \prop_if_exist:NTF \l_stex_current_repository_prop {
592       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
593       \exp_args:Ne \l_tmpa_cs{
594         \prop_item:Nn \l_stex_current_repository_prop { id }
595       }
596     }{
597       \l_tmpa_cs{}
598     }
599   }{
600     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
601     \stex_require_repository:n \l_tmpa_str
602     \str_set:Nx \l_tmpa_str { #1 }
603     \exp_args:Nne \use:nn {
604       \stex_set_current_repository:n \l_tmpa_str
605       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
606     }{
607       \stex_debug:nn{mathhub}{switching~back~to:~
608       \prop_if_exist:NTF \l_stex_current_repository_prop {
609         \prop_item:Nn \l_stex_current_repository_prop { id }::~

```

```

610         \meaning\l_stex_current_repository_prop
611     }{
612         no~repository
613     }
614 }
615 \prop_if_exist:NTF \l_stex_current_repository_prop {
616     \stex_set_current_repository:n {
617         \prop_item:Nn \l_stex_current_repository_prop { id }
618     }
619 }{
620     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
621 }
622 }
623 }
624 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn
625 \newif \ifinputref \inputreffalse
626
627 \cs_new_protected:Nn \stex_mhinput:nn {
628     \stex_in_repository:nn {#1} {
629         \ifinputref
630             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
631         \else
632             \inputreftrue
633             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
634             \inputreffalse
635         \fi
636     }
637 }
638 \NewDocumentCommand \mhinput { 0{} m}{
639     \stex_mhinput:nn{ #1 }{ #2 }
640 }
641
642 \cs_new_protected:Nn \stex_inputref:nn {
643     \stex_in_repository:nn {#1} {
644         \bool_lazy_any:nTF {
645             {\rustex_if_p:} {\latexml_if_p:}
646         } {
647             \str_clear:N \l_tmpa_str
648             \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
649                 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
650             }
651             \stex_annotate_invisible:nnn{inputref}{
652                 \l_tmpa_str / #2
653             }{}
654         }{
655             \begingroup
656             \inputreftrue
657             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
658             \endgroup
659         }

```

```

660 }
661 }
662
663 \NewDocumentCommand \inputref { 0{} m}{
664   \stex_inputref:nn{ #1 }{ #2 }
665 }
666
667 \cs_new_protected:Nn \stex_mhbibresource:nn {
668   \stex_in_repository:nn {#1} {
669     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
670   }
671 }
672 \newcommand\addmhbibresource[2][]{
673   \stex_mhbibresource:nn{ #1 }{ #2 }
674 }

```

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

`\mhpath`

```

675 \def \mhpath #1 #2 {
676   \exp_args:Ne \str_if_eq:nnTF{#1}{-}{
677     \c_stex_mathhub_str /
678     \prop_item:Nn \l_stex_current_repository_prop { id }
679     / source / #2
680   }{
681     \c_stex_mathhub_str / #1 / source / #2
682   }
683 }

```

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

`\libinput`

```

684 \cs_new_protected:Npn \libinput #1 {
685   \prop_if_exist:NF \l_stex_current_repository_prop {
686     \msg_error:nnn{stex}{error/notinarchive}\libinput
687   }
688   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
689     \msg_error:nnn{stex}{error/notinarchive}\libinput
690   }
691   \bool_set_false:N \l_tmpa_bool
692   \tl_clear:N \l_tmpa_tl
693   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
694   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
695   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
696   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
697     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
698     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
699       / meta-inf / lib / #1.tex}{
700       \bool_set_true:N \l_tmpa_bool
701       \tl_put_right:Nx \l_tmpa_tl {
702         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
703           / meta-inf / lib / #1.tex}
704       }
705     }{}

```

```

706 }
707 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
708   / \l_tmpa_str / lib / #1.tex
709 }{
710   \bool_set_true:N \l_tmpa_bool
711   \tl_put_right:Nx \l_tmpa_tl {
712     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
713       / \l_tmpa_str / lib / #1.tex}
714   }
715 }{}
716 \bool_if:NF \l_tmpa_bool {
717   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
718 }
719 \l_tmpa_tl
720 }

```

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

```

721 </package>

```

Chapter 27

STEX -References Implementation

```
722 <*package>
723
724 %%%%%%%%%% references.dtx %%%%%%%%%%
725
726 %\RequirePackage{hyperref}
727 %\RequirePackage{cleveref}
728 <@@=stex_refs>
729
730 Warnings and error messages
731
732 \iow_new:N \c__stex_refs_refs_iow
733 \AddToHook{begindocument}{
734   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
735 }
736 \AddToHook{enddocument}{
737   \iow_close:N \c__stex_refs_refs_iow
738 }
739
740 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
741
742 \NewDocumentCommand \STEXreftitle { m } {
743   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
744 }
745
```

27.1 Document URIs and URLs

```
743 \seq_new:N \g__stex_refs_all_refs_seq
744
745 \str_new:N \l_stex_current_docns_str
746
747 \cs_new_protected:Nn \stex_get_document_uri: {
748   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
749   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
750   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
751   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
752 }
753
```



```

752 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
753
754 \str_clear:N \l_tmpa_str
755 \prop_if_exist:NT \l_stex_current_repository_prop {
756   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
757     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
758   }
759 }
760
761 \str_if_empty:NTF \l_tmpa_str {
762   \str_set:Nx \l_stex_current_docns_str {
763     file:/\stex_path_to_string:N \l_tmpa_seq
764   }
765 }{
766   \bool_set_true:N \l_tmpa_bool
767   \bool_while_do:Nn \l_tmpa_bool {
768     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
769     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
770       {source} { \bool_set_false:N \l_tmpa_bool }
771     }{}{
772       \seq_if_empty:NT \l_tmpa_seq {
773         \bool_set_false:N \l_tmpa_bool
774       }
775     }
776   }
777
778   \seq_if_empty:NTF \l_tmpa_seq {
779     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
780   }{
781     \str_set:Nx \l_stex_current_docns_str {
782       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
783     }
784   }
785 }
786 }
787
788 \str_new:N \l_stex_current_docurl_str
789 \cs_new_protected:Nn \stex_get_document_url: {
790   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
791   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
792   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
793   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
794   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
795
796   \str_clear:N \l_tmpa_str
797   \prop_if_exist:NT \l_stex_current_repository_prop {
798     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
799       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
800         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
801       }
802     }
803
804     \str_if_empty:NTF \l_tmpa_str {
805       \str_set:Nx \l_stex_current_docurl_str {

```

```

806     file:/\stex_path_to_string:N \l_tmpa_seq
807   }
808 }{
809   \bool_set_true:N \l_tmpa_bool
810   \bool_while_do:Nn \l_tmpa_bool {
811     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
812     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
813       {source} { \bool_set_false:N \l_tmpa_bool }
814     }{}{
815       \seq_if_empty:NT \l_tmpa_seq {
816         \bool_set_false:N \l_tmpa_bool
817       }
818     }
819   }
820
821   \seq_if_empty:NTF \l_tmpa_seq {
822     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
823   }{
824     \str_set:Nx \l_stex_current_docurl_str {
825       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
826     }
827   }
828 }
829 }

```

27.2 Setting Reference Targets

```

830 \str_const:Nn \c__stex_refs_url_str{URL}
831 \str_const:Nn \c__stex_refs_ref_str{REF}
832 % @currentlabel -> number
833 % @currentlabelname -> title
834 % @currentHref -> name.number <- id of some kind
835 % \theH# -> \arabic{section}
836 % \the# -> number
837 % \hyper@makecurrent{#}
838 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
839   \stex_get_document_uri:
840   \str_set:Nx \l_tmpa_str { #1 }
841   \str_if_empty:NT \l_tmpa_str {
842     \int_zero:N \l_tmpa_int
843     \bool_set_true:N \l_tmpa_bool
844     \bool_while_do:Nn \l_tmpa_bool {
845       \cs_if_exist:cTF {
846         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
847       }{
848         \int_incr:N \l_tmpa_int
849       }{
850         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
851         \bool_set_false:N \l_tmpa_bool
852       }
853     }
854   }
855   \str_set:Nx \l_tmpa_str {
856     \l_stex_current_docns_str??\l_tmpa_str

```

```

857 }
858 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
859 \stex_if_smsmode:TF {
860   \stex_get_document_url:
861   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
862   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
863 }{
864   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
865   \exp_args:Nx\label{sref_\l_tmpa_str}
866
867   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
868   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
869 }
870 }
871 \cs_new_protected:Npn \stexauxadddocref #1 {
872   \str_set:Nx \l_tmpa_str {#1}
873   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
874   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
875 }
876 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
877   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
878 }

```

27.3 Using References

```

879 \str_new:N \l__stex_refs_indocument_str
880 \keys_define:nn { stex / sref } {
881   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
882   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
883   pre           .tl_set:N = \l__stex_refs_pre_tl ,
884   post          .tl_set:N = \l__stex_refs_post_tl ,
885   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
886 }
887
888 \bool_new:N \c__stex_refs_hyperref_bool
889 \bool_set_false:N \c__stex_refs_hyperref_bool
890 \AddToHook{begindocument}{
891   \@ifpackageloaded{hyperref}{
892     \bool_set_true:N \c__stex_refs_hyperref_bool
893   }{}
894 }
895
896
897 \cs_new_protected:Nn \__stex_refs_args:n {
898   \tl_clear:N \l__stex_refs_linktext_tl
899   \tl_clear:N \l__stex_refs_fallback_tl
900   \tl_clear:N \l__stex_refs_pre_tl
901   \tl_clear:N \l__stex_refs_post_tl
902   \str_clear:N \l__stex_refs_repo_str
903   \keys_set:nn { stex / sref } { #1 }
904 }
905
906 \NewDocumentCommand \sref { 0{} m}{
907   \__stex_refs_args:n { #1 }

```

```

908 \str_if_empty:NTF \l__stex_refs_indocument_str {
909   \str_set:Nn \l_tmpa_str { #2 }
910   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
911   \tl_set:Nn \l_tmpa_tl {
912     \l__stex_refs_fallback_tl
913   }
914   \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
915     \str_set:Nn \l_tmpb_str { ##1 }
916     \str_if_eq:eeT { \l_tmpa_str } {
917       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
918     } {
919       \seq_map_break:n {
920         \tl_set:Nn \l_tmpa_tl {
921           % doc uri in \l_tmpb_str
922           \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
923           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
924             % reference
925             \cs_if_exist:cTF{autoref}{
926               \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
927             }{
928               \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
929             }
930           }{
931             % URL
932             \if_bool:N \c__stex_refs_hyperref_bool {
933               \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
934             }{
935               \l__stex_refs_fallback_tl
936             }
937           }
938         }
939       }
940     }
941   }
942   \l_tmpa_tl
943 }{
944   % TODO
945 }
946 }
947
948 </package>

```

Chapter 28

STEX -Modules Implementation

```
949 <*package>
950
951 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
952
953 <@@=stex_modules>
954
955   Warnings and error messages
956 \msg_new:nnn{stex}{error/unknownmodule}{
957   No~module~#1~found
958 }
959 \msg_new:nnn{stex}{error/syntax}{
960   Syntax~error:~#1
961 }
962 \msg_new:nnn{stex}{error/siglanguage}{
963   Module~#1~declares~signature~#2,~but~does~not~
964   declare~its~language
965 }
966 \msg_new:nnn{stex}{error/conclictingmodules}{
967   Conflicting~imports~for~module~#1
968 }
969
970 \l_stex_current_module_str The current module:
971 \str_new:N \l_stex_current_module_str
972
973 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
974
975 \l_stex_all_modules_seq Stores all available modules
976 \seq_new:N \l_stex_all_modules_seq
977
978 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
979
980 \stex_if_in_module_p:
981 \stex_if_in_module:TF
982 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
983   \str_if_empty:NTF \l_stex_current_module_str
984     \prg_return_false: \prg_return_true:
985 }
```

(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`

```
974 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
975   \prop_if_exist:cTF { c_stex_module_#1_prop }
976   \prg_return_true: \prg_return_false:
977 }
```

(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:

```
978 \cs_new_protected:Nn \stex_add_to_current_module:n {
979   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
980 }
981 \cs_new_protected:Npn \STEXexport {
982   \begingroup
983   \newlinechar=-1\relax
984   \endlinechar=-1\relax
985   %\catcode'\ = 9\relax
986   \expandafter\endgroup\STEXexport:n
987 }
988 \cs_new_protected:Nn \STEXexport:n {
989   \ignorespaces #1
990   \stex_add_to_current_module:n { \ignorespaces #1 }
991   \stex_smsmode_set_codes:
992 }
993 \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`

```
994 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
995   \str_set:Nx \l_tmpa_str { #1 }
996   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
997 }
998
999 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1000 % \str_set:Nx \l_tmpa_str { #1 }
1001 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1002 %}
```

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

`\stex_collect_imports:n`

```
1003 \cs_new_protected:Nn \stex_collect_imports:n {
1004   \seq_clear:N \l_stex_collect_imports_seq
1005   \__stex_modules_collect_imports:n {#1}
1006 }
1007 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1008   \seq_map_inline:cn {c_stex_module_#1_imports} {
1009     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1010       \__stex_modules_collect_imports:n { ##1 }
1011     }
1012 }
```

```

1012 }
1013 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1014   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
1015 }
1016 }

```

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

`\stex_add_import_to_current_module:n`

```

1017 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1018   \str_set:Nx \l_tmpa_str { #1 }
1019   \exp_args:Nno
1020   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1021     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1022   }
1023 }

```

(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).

```

1024 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1025   \str_set:Nx \l_tmpa_str { #1 }
1026   \seq_set_eq:NN \l_tmpa_seq #2
1027   % split off file extension
1028   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1029   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1030   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1031   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1032
1033   \bool_set_true:N \l_tmpa_bool
1034   \bool_while_do:Nn \l_tmpa_bool {
1035     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1036     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1037       {source} { \bool_set_false:N \l_tmpa_bool }
1038     }{}{
1039       \seq_if_empty:NT \l_tmpa_seq {
1040         \bool_set_false:N \l_tmpa_bool
1041       }
1042     }
1043   }
1044
1045   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1046   \str_if_empty:NTF \l_stex_modules_subpath_str {
1047     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1048   }{
1049     \str_set:Nx \l_stex_modules_ns_str {
1050       \l_tmpa_str/\l_stex_modules_subpath_str
1051     }
1052   }
1053 }

```

(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
1054 \str_new:N \l_stex_modules_ns_str
1055 \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.

```

1056 \cs_new_protected:Nn \stex_modules_current_namespace: {
1057   \str_clear:N \l_stex_modules_subpath_str
1058   \prop_if_exist:NTF \l_stex_current_repository_prop {
1059     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1060     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1061   }{
1062     % split off file extension
1063     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1064     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1065     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1066     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1067     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1068     \str_set:Nx \l_stex_modules_ns_str {
1069       file:/\stex_path_to_string:N \l_tmpa_seq
1070     }
1071   }
1072 }

```

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

28.1 The module environment

module arguments:

```

1073 \keys_define:nn { stex / module } {
1074   title      .str_set_x:N = \l_stex_module_title_str ,
1075   ns         .str_set_x:N = \l_stex_module_ns_str ,
1076   lang       .str_set_x:N = \l_stex_module_lang_str ,
1077   sig        .str_set_x:N = \l_stex_module_sig_str ,
1078   creators   .str_set_x:N = \l_stex_module_creators_str ,
1079   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1080   meta       .str_set_x:N = \l_stex_module_meta_str ,
1081   srccite    .str_set_x:N = \l_stex_module_srccite_str
1082 }
1083
1084 \cs_new_protected:Nn \__stex_modules_args:n {
1085   \str_clear:N \l_stex_module_title_str
1086   \str_clear:N \l_stex_module_ns_str
1087   \str_clear:N \l_stex_module_lang_str
1088   \str_clear:N \l_stex_module_sig_str
1089   \str_clear:N \l_stex_module_creators_str
1090   \str_clear:N \l_stex_module_contributors_str
1091   \str_clear:N \l_stex_module_meta_str
1092   \str_clear:N \l_stex_module_srccite_str
1093   \keys_set:nn { stex / module } { #1 }

```



```

1094 }
1095
1096 % module parameters here? In the body?
1097

```

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

```

1098 \cs_new_protected:Nn \stex_module_setup:nn {
1099   \str_set:Nx \l_stex_module_name_str { #2 }
1100   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1101 \stex_if_in_module:TF {
1102   % Nested module
1103   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1104   { ns } \l_stex_module_ns_str
1105   \str_set:Nx \l_stex_module_name_str {
1106     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1107     { name } / \l_stex_module_name_str
1108   }
1109 }{
1110   % not nested:
1111   \str_if_empty:NT \l_stex_module_ns_str {
1112     \stex_modules_current_namespace:
1113     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1114     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1115     / {\l_stex_module_ns_str}
1116     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1117     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1118       \str_set:Nx \l_stex_module_ns_str {
1119         \stex_path_to_string:N \l_tmpa_seq
1120       }
1121     }
1122   }
1123 }

```

Next, we determine the language of the module:

```

1124 \str_if_empty:NT \l_stex_module_lang_str {
1125   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1126   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1127   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1128   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1129   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1130     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1131       inferred~from~file~name}
1132     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1133   }
1134 }
1135
1136 \str_if_empty:NF \l_stex_module_lang_str {
1137   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1138   \l_tmpa_str {
1139     \ltx@ifpackageloaded{babel}{
1140       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1141     }{}
1142   } {
1143     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1144   }
1145 }

```

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

```

1146 \str_if_empty:NTF \l_stex_module_sig_str {
1147   \exp_args:Nnx \prop_gset_from_keyval:cn {
1148     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1149   } {
1150     name      = \l_stex_module_name_str ,
1151     ns        = \l_stex_module_ns_str ,
1152     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1153     lang      = \l_stex_module_lang_str ,
1154     sig       = \l_stex_module_sig_str ,
1155     meta      = \l_stex_module_meta_str
1156   }
1157   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1158   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1159   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1160   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1161   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1162 \str_if_empty:NT \l_stex_module_meta_str {
1163   \str_set:Nx \l_stex_module_meta_str {
1164     \c_stex_metatheory_ns_str ? Metatheory
1165   }
1166 }
1167 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1168   \bool_set_true:N \l_stex_in_meta_bool
1169   \exp_args:Nx \stex_add_to_current_module:n {
1170     \bool_set_true:N \l_stex_in_meta_bool
1171     \stex_activate_module:n {\l_stex_module_meta_str}
1172     \bool_set_false:N \l_stex_in_meta_bool
1173   }
1174   \stex_activate_module:n {\l_stex_module_meta_str}
1175   \bool_set_false:N \l_stex_in_meta_bool
1176 }
1177 }{
1178   \str_if_empty:NT \l_stex_module_lang_str {
1179     \msg_error:nnxx{stex}{error/siglanguage}{
1180       \l_stex_module_ns_str?\l_stex_module_name_str
1181     }{\l_stex_module_sig_str}
1182   }
1183
1184   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1186   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1187   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1188   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1189   \str_set:Nx \l_tmpa_str {

```

```

1190     \stex_path_to_string:N \l_tmpa_seq /
1191     \l_tmpa_str . \l_stex_module_sig_str .tex
1192 }
1193 \IfFileExists \l_tmpa_str {
1194     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1195         \seq_clear:N \l_stex_all_modules_seq
1196         %\prop_clear:N \l_stex_current_module_prop
1197         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1198         \input { \l_tmpa_str }
1199     }
1200 }{
1201     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1202 }
1203 \stex_activate_module:n {
1204     \l_stex_module_ns_str ? \l_stex_module_name_str
1205 }
1206 %\prop_set_eq:Nc \l_stex_current_module_prop {
1207 %   c_stex_module_
1208 %   \l_stex_module_ns_str ?
1209 %   \l_stex_module_name_str
1210 %   _prop
1211 %}
1212 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1213 }
1214 }

```

(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}

1215 \int_new:N \l_stex_module_group_depth_int
1216 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1217     \stex_reactivate_macro:N \STEXexport
1218     \stex_reactivate_macro:N \importmodule
1219     \stex_reactivate_macro:N \symdecl
1220     \stex_reactivate_macro:N \notation
1221     \stex_reactivate_macro:N \symdef
1222     \stex_module_setup:nn{#1}{#2}
1223
1224     \stex_debug:nn{modules}{
1225         New~module:\\
1226         Namespace:~\l_stex_module_ns_str\\
1227         Name:~\l_stex_module_name_str\\
1228         Language:~\l_stex_module_lang_str\\
1229         Signature:~\l_stex_module_sig_str\\
1230         Metatheory:~\l_stex_module_meta_str\\
1231         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1232     }
1233
1234     \seq_put_right:Nx \l_stex_all_modules_seq {
1235         \l_stex_module_ns_str ? \l_stex_module_name_str
1236     }
1237

```

```

1238 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1239 % { \l_stex_module_ns_str ? \l_stex_module_name_str }
1240
1241
1242 \stex_if_smsmode:TF {
1243   \stex_smsmode_set_codes:
1244 } {
1245   \begin{stex_annotate_env} {theory} {
1246     \l_stex_module_ns_str ? \l_stex_module_name_str
1247   }
1248
1249   \stex_annotate_invisible:nnn{header}{} {
1250     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1251     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1252     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1253       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1254     }
1255   }
1256 }
1257 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1258 % TODO: Inherit metatheory for nested modules?
1259 }
1260 \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}

1261 \cs_new_protected:Nn \_stex_modules_end_module: {
1262 % \str_set:Nx \l_tmpa_str {
1263 %   c_stex_module_
1264 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1265 %   \prop_item:Nn \l_stex_current_module_prop { name }
1266 %   _prop
1267 % }
1268 %^^A \prop_new:c { \l_tmpa_str }
1269 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1270 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1271 }

```

(End definition for _stex_modules_end_module:.)

@module The core environment, with no header

```

1272 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1273 \NewDocumentEnvironment { @module } { 0 } { m } {
1274   \par
1275   \_stex_modules_begin_module:nn{#1}{#2}
1276 } {
1277   \_stex_modules_end_module:
1278   \stex_if_smsmode:TF {
1279 % \exp_args:Nx \stex_add_to_sms:n {
1280 %   \prop_gset_from_keyval:cn {
1281 %     c_stex_module_
1282 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1283 %     \prop_item:Nn \l_stex_current_module_prop { name }

```

```

1284 %      _prop
1285 %      } {
1286 %      name      = \prop_item:cn { \l_tmpa_str } { name } ,
1287 %      ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1288 %      file      = \prop_item:cn { \l_tmpa_str } { file } ,
1289 %      lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1290 %      sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1291 %      meta      = \prop_item:cn { \l_tmpa_str } { meta }
1292 %      }
1293 %      }
1294 }{
1295   \end{stex_annotate_env}
1296 }
1297 }

```

\stex_modules_heading: Code for document headers

```

1298 \cs_if_exist:NTF \thesection {
1299   \newcounter{module}[section]
1300 }{
1301   \newcounter{module}
1302 }
1303
1304 \bool_if:NT \c_stex_showmods_bool {
1305   \latexml_if:F { \RequirePackage{mdframed} }
1306 }
1307
1308 \cs_new_protected:Nn \stex_modules_heading: {
1309   \stepcounter{module}
1310   \par
1311   \bool_if:NT \c_stex_showmods_bool {
1312     \noindent{\textbf{Module} ~
1313       \cs_if_exist:NTF \thesection {\thesection.}
1314       \themodule ~ [\l_stex_module_name_str]
1315     }
1316     \str_if_empty:NTF \l_stex_module_title_str {
1317       }{
1318         \quad(\l_stex_module_title_str)\hfill
1319       }\par
1320     }
1321     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1322     % TODO
1323     \stex_ref_new_doc_target:n \l_stex_module_name_str
1324   }

```

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

Finally:

```

1325 \NewDocumentEnvironment { module } { 0 } { m } {
1326   \bool_if:NT \c_stex_showmods_bool {
1327     \begin{mdframed}
1328   }
1329   \begin{@module}[#1]{#2}
1330     \stex_modules_heading:
1331   }{
1332     \end{@module}

```

```

1333 \bool_if:NT \c_stex_showmods_bool {
1334   \end{mdframed}
1335 }
1336 }

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1337 \NewDocumentCommand \STEXModule { m } {
1338   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1339   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1340   \tl_set:Nn \l_tmpa_tl {
1341     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1342   }
1343   \seq_map_inline:Nn \l_stex_all_modules_seq {
1344     \str_set:Nn \l_tmpb_str { ##1 }
1345     \str_if_eq:eeT { \l_tmpa_str } {
1346       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1347     } {
1348       \seq_map_break:n {
1349         \tl_set:Nn \l_tmpa_tl {
1350           \stex_invoke_module:n { ##1 }
1351         }
1352       }
1353     }
1354   }
1355   \l_tmpa_tl
1356 }
1357
1358 \cs_new_protected:Nn \stex_invoke_module:n {
1359   \stex_debug:nn{modules}{Invoking~module~#1}
1360   \peek_charcode_remove:NTF ! {
1361     \__stex_modules_invoke_uri:nN { #1 }
1362   } {
1363     \peek_charcode_remove:NTF ? {
1364       \__stex_modules_invoke_symbol:nn { #1 }
1365     } {
1366       \msg_error:nnx{stex}{error/syntax}{
1367         ?~or~!~expected~after~
1368         \c_backslash_str STEXModule{#1}
1369       }
1370     }
1371   }
1372 }
1373
1374 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1375   \str_set:Nn #2 { #1 }
1376 }
1377
1378 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1379   \stex_invoke_symbol:n{#1?#2}
1380 }

```

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

`\stex_activate_module:n`

```

1381 \bool_new:N \l_stex_in_meta_bool
1382 \bool_set_false:N \l_stex_in_meta_bool
1383 \cs_new_protected:Nn \stex_activate_module:n {
1384   \stex_debug:nn{modules}{Activating~module~#1}
1385   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1386     \msg_error:nnn{stex}{error/concllictingmodules}{ #1 }
1387   }
1388   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1389     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1390     \use:c{ c_stex_module_#1_code }
1391   }
1392 }

```

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

```

1393 </package>

```

Chapter 29

sTeX -Module Inheritance Implementation

```
1394 <*package>
1395
1396 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1397
```

29.1 SMS Mode

```
1398 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1399 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1400 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1401 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1402
1403 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1404   \makeatletter
1405   \makeatother
1406   \ExplSyntaxOn
1407   \ExplSyntaxOff
1408 }
1409
1410 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1411   \symdef
1412   \importmodule
1413   \notation
1414   \symdecl
1415   \STEXexport
1416 }
1417
1418 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1419   \tl_to_str:n {
1420     module,
1421     @module
```



```

1422 }
1423 }

```

(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
1424 \bool_new:N \g__stex_smsmode_bool
1425 \bool_set_false:N \g__stex_smsmode_bool
1426 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1427   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1428 }

```

(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
1429 \bool_new:N \g__stex_smsmode_catcode_bool
1430 \bool_set_false:N \g__stex_smsmode_catcode_bool
1431 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1432   \bool_if:NTF \g__stex_smsmode_catcode_bool
1433   \prg_return_true: \prg_return_false:
1434 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:
1435 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1436   \stex_if_smsmode:T {
1437     \__stex_smsmode_if_catcodes:F {
1438       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1439       \exp_after:wN \char_gset_active_eq:NN
1440       \c_backslash_str \__stex_smsmode_cs:
1441       \tex_global:D \char_set_catcode_active:N \
1442       \tex_global:D \char_set_catcode_other:N $
1443       \tex_global:D \char_set_catcode_other:N ^
1444       \tex_global:D \char_set_catcode_other:N _
1445       \tex_global:D \char_set_catcode_other:N &
1446       \tex_global:D \char_set_catcode_other:N ##
1447     }
1448   }
1449 } \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.
1450 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1451   \__stex_smsmode_if_catcodes:T {
1452     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1453     \exp_after:wN \tex_global:D \exp_after:wN
1454     \char_set_catcode_escape:N \c_backslash_str
1455     \tex_global:D \char_set_catcode_math_toggle:N $
1456     \tex_global:D \char_set_catcode_math_superscript:N ^
1457     \tex_global:D \char_set_catcode_math_subscript:N _
1458     \tex_global:D \char_set_catcode_alignment:N &
1459     \tex_global:D \char_set_catcode_parameter:N ##
1460   }
1461 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1462 \cs_new_protected:Nn \stex_in_smsmode:nn {
1463   \vbox_set:Nn \l_tmpa_box {
1464     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1465     \bool_gset_true:N \g__stex_smsmode_bool
1466     \stex_smsmode_set_codes:
1467     #2
1468     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1469     \stex_if_smsmode:F {
1470       \__stex_smsmode_unset_codes:
1471     }
1472   }
1473   \box_clear:N \l_tmpa_box
1474 }

```

(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:

```

1475 \cs_new_protected:Nn \_stex_smsmode_cs: {
1476   \str_clear:N \l_tmpa_str
1477   \peek_analysis_map_inline:n {
1478     % #1: token (one expansion)
1479     % #2: charcode
1480     % #3 catcode
1481     \token_if_eq_charcode:NNTF ##3 B {
1482       % token is a letter
1483       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1484     } {
1485       \str_if_empty:NTF \l_tmpa_str {
1486         % we don't allow (or need) single non-letter CSs
1487         % for now
1488         \peek_analysis_map_break:
1489       }{
1490         \str_if_eq:onTF \l_tmpa_str { begin } {
1491           \peek_analysis_map_break:n {
1492             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1493           }
1494         } {
1495           \str_if_eq:onTF \l_tmpa_str { end } {
1496             \peek_analysis_map_break:n {
1497               \exp_after:wN \_stex_smsmode_checkend:n ##1
1498             }
1499           } {
1500             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1501             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1502               \g_stex_smsmode_allowedmacros_tl
1503               { \use:c{\l_tmpa_str} } {
1504               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1505               \peek_analysis_map_break:n {
1506                 \exp_after:wN \l_tmpa_tl ##1
1507               }
1508             }
1509           }
1510         }
1511       }
1512     }
1513   }
1514 }

```

```

1508     } {
1509         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1510         \g_stex_smsmode_allowedmacros_escape_tl
1511         { \use:c{\l_tmpa_str} } {
1512             \__stex_smsmode_unset_codes:
1513             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1514             % TODO \__stex_smsmode_rescan_cs:
1515             % \int_compare:nNnTF {##2} = {92} {
1516             %     \peek_analysis_map_break:n {
1517             %         \__stex_smsmode_unset_codes:
1518             %         \__stex_smsmode_rescan_cs:
1519             %     }
1520             % } {
1521             %     \peek_analysis_map_break:n {
1522             %         \exp_after:wN \l_tmpa_tl ##1
1523             %     }
1524             % }
1525             } {
1526                 \int_compare:nNnTF {##2} = {92} {
1527                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1528                 }{
1529                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1530                 }
1531             }
1532         }
1533     }
1534 }
1535 }
1536 }
1537 }
1538 }

```

(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:

```

1539 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1540     \str_clear:N \l_tmpb_str
1541     \peek_analysis_map_inline:n {
1542         \token_if_eq_charcode:NNTF ##3 B {
1543             % token is a letter
1544             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1545         } {
1546             \peek_analysis_map_break:n {
1547                 \exp_after:wN \use:c \exp_after:wN {
1548                     \exp_after:wN \l_tmpa_str\exp_after:wN
1549                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1550             }
1551         }
1552     }
1553 }

```

(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.

```

1554 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1555   \str_set:Nn \l_tmpa_str { #1 }
1556   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1557     \__stex_smsmode_unset_codes:
1558     \begin{#1}
1559   }
1560 }
```

(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.

```

1561 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1562   \str_set:Nn \l_tmpa_str { #1 }
1563   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1564     \end{#1}
1565   }
1566 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1567 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1568 \cs_new_protected:Nn \stex_import_module_uri:nn {
1569   \str_set:Nx \l_stex_import_archive_str { #1 }
1570   \str_set:Nn \l_stex_import_path_str { #2 }
1571
1572   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1573   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1574   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1575
1576   \stex_modules_current_namespace:
1577   \bool_lazy_all:nTF {
1578     {\str_if_empty_p:N \l_stex_import_archive_str}
1579     {\str_if_empty_p:N \l_stex_import_path_str}
1580     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1581   }{
1582     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1583     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1584   }{
1585     \str_if_empty:NT \l_stex_import_archive_str {
1586       \prop_if_exist:NT \l_stex_current_repository_prop {
1587         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1588       }
1589     }
1590     \str_if_empty:NTF \l_stex_import_archive_str {
1591       \str_if_empty:NF \l_stex_import_path_str {
1592         \str_set:Nx \l_stex_import_ns_str {
1593           \l_stex_module_ns_str / \l_stex_import_path_str
1594         }
1595       }
1596     }
```

```

1596   }{
1597     \stex_require_repository:n \l_stex_import_archive_str
1598     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
1599     \l_stex_import_ns_str
1600     \str_if_empty:NF \l_stex_import_path_str {
1601       \str_set:Nx \l_stex_import_ns_str {
1602         \l_stex_import_ns_str / \l_stex_import_path_str
1603       }
1604     }
1605   }
1606 }
1607 }

```

(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 1608 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1609 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1610 \str_new:N \l_stex_import_path_str
                           1611 \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>}
1612 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1613   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1614
1615     % archive
1616     \str_set:Nx \l_tmpa_str { #2 }
1617     \str_if_empty:NTF \l_tmpa_str {
1618       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1619     } {
1620       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1621       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1622       \seq_put_right:Nn \l_tmpa_seq { source }
1623     }
1624
1625     % path
1626     \str_set:Nx \l_tmpb_str { #3 }
1627     \str_if_empty:NTF \l_tmpb_str {
1628       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1629
1630       \ltx@ifpackageloaded{babel} {
1631         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1632           { \language } \l_tmpb_str {
1633           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1634         }
1635       } {
1636         \str_clear:N \l_tmpb_str
1637       }
1638
1639       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1640       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1641         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1642     }{
1643       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1644       \IfFileExists{ \l_tmpa_str.tex }{
1645         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1646       }{
1647         % try english as default
1648         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1649         \IfFileExists{ \l_tmpa_str.en.tex }{
1650           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1651         }{
1652           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1653         }
1654       }
1655     }
1656
1657   } {
1658     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1659     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1660
1661     \ltx@ifpackageloaded{babel} {
1662       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1663       { \language } \l_tmpb_str {
1664         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1665       }
1666     } {
1667       \str_clear:N \l_tmpb_str
1668     }
1669
1670     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1671
1672     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1673     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1674       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1675     }{
1676       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1677       \IfFileExists{ \l_tmpa_str/#4.tex }{
1678         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1679       }{
1680         % try english as default
1681         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1682         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1683           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1684         }{
1685           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1686           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1687             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1688           }{
1689             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1690             \IfFileExists{ \l_tmpa_str.tex }{
1691               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1692             }{
1693               % try english as default
1694               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1695               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1696         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1697     }{
1698         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1699     }
1700 }
1701 }
1702 }
1703 }
1704 }
1705 }
1706
1707 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1708     \seq_clear:N \l_stex_all_modules_seq
1709     \str_clear:N \l_stex_current_module_str
1710     \str_set:Nx \l_tmpb_str { #2 }
1711     \str_if_empty:NF \l_tmpb_str {
1712         \stex_set_current_repository:n { #2 }
1713     }
1714     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1715     \input { \g__stex_importmodule_file_str }
1716 }
1717
1718 \stex_if_module_exists:nF { #1 ? #4 } {
1719     \msg_error:nnx{stex}{error/unknownmodule}{
1720         #1?#4~(in~file~\g__stex_importmodule_file_str)
1721     }
1722 }
1723 }
1724 \stex_activate_module:n { #1 ? #4 }
1725 }

```

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

`\importmodule`

```

1726 \NewDocumentCommand \importmodule { 0{} m } {
1727     \stex_import_module_uri:nn { #1 } { #2 }
1728     \stex_debug:nn{modules}{Importing~module:~
1729         \l_stex_import_ns_str ? \l_stex_import_name_str
1730     }
1731     \stex_if_smsmode:F {
1732         \stex_import_require_module:nnnn
1733         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1734         { \l_stex_import_path_str } { \l_stex_import_name_str }
1735         \stex_annotate_invisible:nnn
1736         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1737     }
1738     \exp_args:Nx \stex_add_to_current_module:n {
1739         \stex_import_require_module:nnnn
1740         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1741         { \l_stex_import_path_str } { \l_stex_import_name_str }
1742     }
1743     \exp_args:Nx \stex_add_import_to_current_module:n {
1744         \l_stex_import_ns_str ? \l_stex_import_name_str
1745     }

```

```

1746 \stex_smsmode_set_codes:
1747 }
1748 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1749 \NewDocumentCommand \usemodule { 0{} m } {
1750 \stex_if_smsmode:F {
1751 \stex_import_module_uri:nn { #1 } { #2 }
1752 \stex_import_require_module:nnnn
1753 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1754 { \l_stex_import_path_str } { \l_stex_import_name_str }
1755 \stex_annotate_invisible:nnn
1756 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1757 }
1758 \stex_smsmode_set_codes:
1759 }

```

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

```

1760 \</package>

```


Chapter 30

STEX -Symbols Implementation

```
1761 <*package>
1762
1763 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1764
```

Warnings and error messages

```
1765
```

30.1 Symbol Declarations

```
1766 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1767 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1768 \NewDocumentCommand \STEXsymbol { m } {
1769   \stex_get_symbol:n { #1 }
1770   \exp_args:No
1771   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1772 }
```

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

symdecl arguments:

```
1773 \keys_define:nn { stex / symdecl } {
1774   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1775   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1776   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1777   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1778   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1779   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1780   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1781   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1782 }
```

```

1783
1784 \bool_new:N \l_stex_symdecl_make_macro_bool
1785
1786 \cs_new_protected:Nn \__stex_symdecl_args:n {
1787   \str_clear:N \l_stex_symdecl_name_str
1788   \str_clear:N \l_stex_symdecl_args_str
1789   \bool_set_false:N \l_stex_symdecl_local_bool
1790   \tl_clear:N \l_stex_symdecl_type_tl
1791   \tl_clear:N \l_stex_symdecl_definiens_tl
1792
1793   \keys_set:nn { stex / symdecl } { #1 }
1794 }

```

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

```

1795
1796 \NewDocumentCommand \symdecl { s O{} m } {
1797   \__stex_symdecl_args:n { #2 }
1798   \IfBooleanTF #1 {
1799     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1800   } {
1801     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1802   }
1803   \stex_symdecl_do:n { #3 }
1804   \stex_smsmode_set_codes:
1805 }
1806 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1807 \cs_new_protected:Nn \stex_symdecl_do:n {
1808   \stex_if_in_module:F {
1809     % TODO throw error? some default namespace?
1810   }
1811
1812   \str_if_empty:NT \l_stex_symdecl_name_str {
1813     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1814   }
1815
1816   \prop_if_exist:cT { l_stex_symdecl_
1817     \l_stex_current_module_str ?
1818     \l_stex_symdecl_name_str
1819     _prop
1820   }{
1821     % TODO throw error (beware of circular dependencies)
1822   }
1823
1824   \prop_clear:N \l_tmpa_prop
1825   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1826   \seq_clear:N \l_tmpa_seq
1827   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1828   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1829

```

```

1830 \exp_args:No \stex_add_constant_to_current_module:n {
1831   \l_stex_symdecl_name_str
1832 }
1833
1834 % arity/args
1835 \int_zero:N \l_tmpb_int
1836
1837 \bool_set_true:N \l_tmpa_bool
1838 \str_map_inline:Nn \l_stex_symdecl_args_str {
1839   \token_case_meaning:NnF ##1 {
1840     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1841     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1842     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1843     {\tl_to_str:n a} {
1844       \bool_set_false:N \l_tmpa_bool
1845       \int_incr:N \l_tmpb_int
1846     }
1847     {\tl_to_str:n B} {
1848       \bool_set_false:N \l_tmpa_bool
1849       \int_incr:N \l_tmpb_int
1850     }
1851   }{
1852     \msg_set:nnn{stex}{error/wrongargs}{
1853       args~value~in~symbol~declaration~for~
1854       \l_stex_current_module_str ?
1855       \l_stex_symdecl_name_str ~
1856       needs~to~be~
1857       i,~a,~b~or~B,~but~##1~given
1858     }
1859     \msg_error:nn{stex}{error/wrongargs}
1860   }
1861 }
1862 \bool_if:NTF \l_tmpa_bool {
1863   % possibly numeric
1864   \str_if_empty:NTF \l_stex_symdecl_args_str {
1865     \prop_put:Nnn \l_tmpa_prop { args } {}
1866     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1867   }{
1868     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1869     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1870     \str_clear:N \l_tmpa_str
1871     \int_step_inline:nn \l_tmpa_int {
1872       \str_put_right:Nn \l_tmpa_str i
1873     }
1874     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1875   }
1876 } {
1877   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1878   \prop_put:Nnx \l_tmpa_prop { arity }
1879   { \str_count:N \l_stex_symdecl_args_str }
1880 }
1881 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1882
1883

```

```

1884 % semantic macro
1885
1886 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1887   \exp_args:Nx \stex_do_aftergroup:n {
1888     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1889       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1890     }}
1891   }
1892
1893   \bool_if:NF \l_stex_symdecl_local_bool {
1894     \exp_args:Nx \stex_add_to_current_module:n {
1895       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1896         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1897       } }
1898     }
1899   }
1900 }
1901
1902 % add to all symbols
1903
1904 \bool_if:NF \l_stex_symdecl_local_bool {
1905   \exp_args:Nx \stex_add_to_current_module:n {
1906     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1907       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1908     }
1909   }
1910 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1911 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1912 %   }
1913 }
1914
1915 \stex_debug:nn{symbols}{New~symbol:~
1916   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1917   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1918   Args:~\prop_item:Nn \l_tmpa_prop { args }
1919 }
1920
1921 % circular dependencies require this:
1922
1923 \prop_if_exist:cF {
1924   l_stex_symdecl_
1925   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1926   _prop
1927 } {
1928   \prop_set_eq:cN {
1929     l_stex_symdecl_
1930     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1931     _prop
1932   } \l_tmpa_prop
1933 }
1934
1935 \seq_clear:c {
1936   l_stex_symdecl_
1937   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

1938   _notations
1939 }
1940
1941 \bool_if:NF \l_stex_symdecl_local_bool {
1942   \exp_args:Nx
1943   \stex_add_to_current_module:n {
1944     \seq_clear:c {
1945       l_stex_symdecl_
1946       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1947       _notations
1948     }
1949     \prop_set_from_keyval:cn {
1950       l_stex_symdecl_
1951       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1952       _prop
1953     } {
1954       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1955       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1956       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1957       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1958       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1959       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1960     }
1961   }
1962 }
1963
1964 \stex_if_smsmode:TF {
1965   \bool_if:NF \l_stex_symdecl_local_bool {
1966     % \exp_args:Nx \stex_add_to_sms:n {
1967     %   \prop_set_from_keyval:cn {
1968     %     l_stex_symdecl_
1969     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1970     %     _prop
1971     %   } {
1972     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1973     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1974     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1975     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1976     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1977     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1978     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1979     %   }
1980     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1981     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1982     %   }
1983     % }
1984   }
1985 }{
1986   \exp_args:Nx \stex_do_aftergroup:n {
1987     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1988       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1989     }
1990   }
1991   \stex_if_do_html:T {

```

```

1992 \stex_annotate_invisible:nnn {symdecl} {
1993   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1994 } {
1995   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
1996   \stex_annotate_invisible:nnn{args}{}{
1997     \prop_item:Nn \l_tmpa_prop { args }
1998   }
1999   \stex_annotate_invisible:nnn{macroname}{#1}{}
2000   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2001     \stex_annotate_invisible:nnn{definiens}{}
2002     { $\l_stex_symdecl_definiens_tl$ }
2003   }
2004 }
2005 }
2006 }
2007 }

```

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

`\stex_get_symbol:n`

```

2008 \str_new:N \l_stex_get_symbol_uri_str
2009
2010 \cs_new_protected:Nn \stex_get_symbol:n {
2011   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2012     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2013   }{
2014     % argument is a string
2015     % is it a command name?
2016     \cs_if_exist:cTF { #1 }{
2017       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2018       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2019       \str_if_empty:NNTF \l_tmpa_str {
2020         \exp_args:Nx \cs_if_eq:NNTF {
2021           \tl_head:N \l_tmpa_tl
2022         } \stex_invoke_symbol:n {
2023           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2024         }{
2025           \__stex_symdecl_get_symbol_from_string:n { #1 }
2026         }
2027       } {
2028         \__stex_symdecl_get_symbol_from_string:n { #1 }
2029       }
2030     }{
2031       % argument is not a command name
2032       \__stex_symdecl_get_symbol_from_string:n { #1 }
2033       % \l_stex_all_symbols_seq
2034     }
2035   }
2036 }
2037
2038 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2039   \str_set:Nn \l_tmpa_str { #1 }
2040   \bool_set_false:N \l_tmpa_bool
2041   \stex_if_in_module:T {

```

```

2042 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2043 \bool_set_true:N \l_tmpa_bool
2044 \str_set:Nx \l_stex_get_symbol_uri_str {
2045 \l_stex_current_module_str ? #1
2046 }
2047 }
2048 }
2049 \bool_if:NF \l_tmpa_bool {
2050 \tl_set:Nn \l_tmpa_tl {
2051 \msg_set:nnn{stex}{error/unknownsymbol}{
2052 No~symbol~#1~found!
2053 }
2054 \msg_error:nn{stex}{error/unknownsymbol}
2055 }
2056 \str_set:Nn \l_tmpa_str { #1 }
2057 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2058 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2059 \str_set:Nn \l_tmpb_str { ##1 }
2060 \str_if_eq:eeT { \l_tmpa_str } {
2061 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2062 } {
2063 \seq_map_break:n {
2064 \tl_set:Nn \l_tmpa_tl {
2065 \str_set:Nn \l_stex_get_symbol_uri_str {
2066 ##1
2067 }
2068 }
2069 }
2070 }
2071 }
2072 \l_tmpa_tl
2073 }
2074 }
2075
2076 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2077 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2078 { \tl_tail:N \l_tmpa_tl }
2079 \tl_if_single:NTF \l_tmpa_tl {
2080 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2081 \exp_after:wN \str_set:Nn \exp_after:wN
2082 \l_stex_get_symbol_uri_str \l_tmpa_tl
2083 }{
2084 % TODO
2085 % tail is not a single group
2086 }
2087 }{
2088 % TODO
2089 % tail is not a single group
2090 }
2091 }

```

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

30.2 Notations

```

2092 <@@=stex_notation>

      notation arguments:
2093 \keys_define:nn { stex / notation } {
2094   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2095   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2096   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2097   op        .tl_set:N   = \l__stex_notation_op_tl ,
2098   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2099   primary   .default:n   = {true} ,
2100   unknown   .code:n      = \str_set:Nx
2101             \l__stex_notation_variant_str \l_keys_key_str
2102 }
2103
2104 \cs_new_protected:Nn \stex_notation_args:n {
2105   \str_clear:N \l__stex_notation_lang_str
2106   \str_clear:N \l__stex_notation_variant_str
2107   \str_clear:N \l__stex_notation_prec_str
2108   \tl_clear:N \l__stex_notation_op_tl
2109   \bool_set_false:N \l__stex_notation_primary_bool
2110
2111   \keys_set:nn { stex / notation } { #1 }
2112 }

```

\notation

```

2113 \NewDocumentCommand \notation { 0{ } m } {
2114   \stex_notation_args:n { #1 }
2115   \tl_clear:N \l_stex_symdecl_definiens_tl
2116   \stex_get_symbol:n { #2 }
2117   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2118 }
2119 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2120 \cs_new_protected:Nn \stex_notation_do:nn {
2121   \let\l_stex_current_symbol_str\relax
2122   \prop_set_eq:Nc \l_tmpa_prop {
2123     l_stex_symdecl_ #1 _prop
2124   }
2125
2126   \prop_clear:N \l_tmpb_prop
2127   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2128   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2129   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2130
2131   % precedences
2132   \seq_clear:N \l_tmpb_seq
2133   \exp_args:NNno
2134   \str_if_empty:NTF \l__stex_notation_prec_str {
2135     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2136     \int_compare:nNnTF \l_tmpa_str = 0 {

```



```

2137     \exp_args:NNx
2138     \prop_put:Nno \l_tmpb_prop { opprec }
2139     { \neginfprec }
2140   }{
2141     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2142   }
2143 } {
2144   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2145     \exp_args:NNx
2146     \prop_put:Nno \l_tmpb_prop { opprec }
2147     { \neginfprec }
2148     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2149     \int_step_inline:nn { \l_tmpa_str } {
2150       \exp_args:NNx
2151       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2152     }
2153   }{
2154     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2155     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2156       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2157       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2158         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2159         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2160         \seq_map_inline:Nn \l_tmpa_seq {
2161           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2162         }
2163       }
2164       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2165     }{
2166       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2167       \int_compare:nNnTF \l_tmpa_str = 0 {
2168         \exp_args:NNx
2169         \prop_put:Nno \l_tmpb_prop { opprec }
2170         { \infprec }
2171       }{
2172         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2173       }
2174     }
2175   }
2176 }
2177
2178 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2179 \int_step_inline:nn { \l_tmpa_str } {
2180   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2181     \exp_args:NNx
2182     \seq_put_right:Nn \l_tmpb_seq {
2183       \prop_item:Nn \l_tmpb_prop { opprec }
2184     }
2185   }
2186 }
2187
2188 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2189 \tl_clear:N \l_tmpa_tl
2190

```

```

2191 \int_compare:nNnTF \l_tmpa_str = 0 {
2192   \exp_args:NNe
2193   \cs_set:Npn \l__stex_notation_macrocode_cs {
2194     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2195     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2196     { \prop_item:Nn \l_tmpb_prop { opprec } }
2197     { \exp_not:n { #2 } }
2198   }
2199   \__stex_notation_final:
2200 }{
2201   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2202   \str_if_in:NnTF \l_tmpb_str b {
2203     \exp_args:Nne \use:nn
2204     {
2205       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2206       \cs_set:Npn \l_tmpa_str } { {
2207         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2208         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2209         { \prop_item:Nn \l_tmpb_prop { opprec } }
2210         { \exp_not:n { #2 } }
2211       } }
2212   }{
2213     \str_if_in:NnTF \l_tmpb_str B {
2214       \exp_args:Nne \use:nn
2215       {
2216         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2217         \cs_set:Npn \l_tmpa_str } { {
2218           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2219           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2220           { \prop_item:Nn \l_tmpb_prop { opprec } }
2221           { \exp_not:n { #2 } }
2222         } }
2223     }{
2224       \exp_args:Nne \use:nn
2225       {
2226         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2227         \cs_set:Npn \l_tmpa_str } { {
2228           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2229           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2230           { \prop_item:Nn \l_tmpb_prop { opprec } }
2231           { \exp_not:n { #2 } }
2232         } }
2233     }
2234   }
2235
2236   \int_zero:N \l_tmpa_int
2237   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2238   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2239   \__stex_notation_arguments:
2240 }
2241 }

```

(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

```

2242 \cs_new_protected:Nn \_stex_notation_arguments: {
2243   \int_incr:N \l_tmpa_int
2244   \str_if_empty:NTF \l_tmpa_str {
2245     \_stex_notation_final:
2246   }{
2247     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2248     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2249     \str_if_eq:VnTF \l_tmpb_str a {
2250       \_stex_notation_argument_assoc:n
2251     }{
2252       \str_if_eq:VnTF \l_tmpb_str B {
2253         \_stex_notation_argument_assoc:n
2254       }{
2255         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2256         \tl_put_right:Nx \l_tmpa_tl {
2257           { \_stex_term_math_arg:nnn
2258             { \int_use:N \l_tmpa_int }
2259             { \l_tmpb_str }
2260             { ####\int_use:N \l_tmpa_int }
2261           }
2262         }
2263         \_stex_notation_arguments:
2264       }
2265     }
2266   }
2267 }
```

(End definition for _stex_notation_arguments:.)

`_stex_notation_argument_assoc:n`

```

2268 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2269   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2270   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2271   \tl_put_right:Nx \l_tmpa_tl {
2272     { \_stex_term_math_assoc_arg:nnnn
2273       { \int_use:N \l_tmpa_int }
2274       { \l_tmpb_str }
2275       \exp_args:No \exp_not:n
2276       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2277       { ####\int_use:N \l_tmpa_int }
2278     }
2279   }
2280   \_stex_notation_arguments:
2281 }
```

(End definition for _stex_notation_argument_assoc:n.)

`_stex_notation_final:` Called after processing all notation arguments

```

2282 \cs_new_protected:Nn \_stex_notation_final: {
2283   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2284   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2285   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2286   \exp_args:Nne \use:nn
```

```

2287 {
2288 \cs_generate_from_arg_count:cNnn {
2289   stex_notation_ \l_tmpa_str \c_hash_str
2290   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2291   _cs
2292 }
2293 \cs_set:Npn \l_tmpb_str } { {
2294   \exp_after:wN \exp_after:wN \exp_after:wN
2295   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2296   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2297 } }
2298
2299 \tl_if_empty:NF \l__stex_notation_op_tl {
2300   \cs_set:cpx {
2301     stex_op_notation_ \l_tmpa_str \c_hash_str
2302     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2303     _cs
2304   } {
2305     \_stex_term_oms:nnn {
2306       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2307       \l__stex_notation_lang_str
2308     }{
2309       \l_tmpa_str
2310     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2311   }
2312 }
2313
2314 \exp_args:Ne
2315 \stex_add_to_current_module:n {
2316   \cs_generate_from_arg_count:cNnn {
2317     stex_notation_ \l_tmpa_str \c_hash_str
2318     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2319     _cs
2320   } \cs_set:Npn {\l_tmpb_str} {
2321     \exp_after:wN \exp_after:wN \exp_after:wN
2322     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2323     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2324   }
2325   \tl_if_empty:NF \l__stex_notation_op_tl {
2326     \cs_set:cpn {
2327       stex_op_notation_ \l_tmpa_str \c_hash_str
2328       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2329       _cs
2330     } {
2331       \_stex_term_oms:nnn {
2332         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2333         \l__stex_notation_lang_str
2334       }{
2335         \l_tmpa_str
2336       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2337     }
2338   }
2339 }
2340

```

```

2341 \seq_put_right:cx {
2342   l_stex_symdecl_
2343   \prop_item:Nn \l_tmpb_prop { symbol }
2344   _notations
2345 } {
2346   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2347 }
2348
2349 \stex_debug:nn{symbols}{
2350   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2351   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2352   Operator~precedence:~
2353   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2354   Argument~precedences:~
2355   \seq_use:Nn \l_tmpa_seq {,~}^^J
2356   Notation: \cs_meaning:c {
2357     stex_notation_ \l_tmpa_str \c_hash_str
2358     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2359     _cs
2360   }
2361 }
2362
2363 \prop_set_eq:cN {
2364   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2365   \c_hash_str \l__stex_notation_lang_str _prop
2366 } \l_tmpb_prop
2367
2368 \exp_args:Ne
2369 \stex_add_to_current_module:n {
2370   \seq_put_right:cn {
2371     l_stex_symdecl_
2372     \prop_item:Nn \l_tmpb_prop { symbol }
2373     _notations
2374   } {
2375     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2376   }
2377   \prop_set_from_keyval:cn {
2378     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2379     \c_hash_str \l__stex_notation_lang_str _prop
2380   } {
2381     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2382     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2383     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2384     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2385     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2386   }
2387 }
2388
2389 \stex_if_smsmode:TF {
2390   \stex_smsmode_set_codes:
2391   % \exp_args:Nx \stex_add_to_sms:n {
2392   %   \prop_set_from_keyval:cn {
2393   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2394   %     \c_hash_str \l__stex_notation_lang_str _prop

```

```

2395 %      } {
2396 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2397 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2398 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2399 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2400 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2401 %      }
2402 %  }
2403 }{
2404
2405 % HTML annotations
2406 \stex_if_do_html:T {
2407     \stex_annotate_invisible:nnn { notation }
2408     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2409         \stex_annotate_invisible:nnn { notationfragment }
2410         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2411         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2412         \stex_annotate_invisible:nnn { precedence }
2413         { \prop_item:Nn \l_tmpb_prop { opprec };
2414           \seq_use:Nn \l_tmpa_seq { x }
2415         }{}
2416
2417         \int_zero:N \l_tmpa_int
2418         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2419         \tl_clear:N \l_tmpa_tl
2420         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2421             \int_incr:N \l_tmpa_int
2422             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2423             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2424             \str_if_eq:VnTF \l_tmpb_str a {
2425                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2426                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2427                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2428                 } }
2429             }{
2430                 \str_if_eq:VnTF \l_tmpb_str B {
2431                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2432                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2433                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2434                     } }
2435                 }{
2436                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2437                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2438                     } }
2439                 }
2440             }
2441         }
2442         \stex_annotate_invisible:nnn { notationcomp }{}{
2443             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2444             $ \exp_args:Nno \use:nn { \use:c {
2445                 stex_notation_ \l_stex_current_symbol_str
2446                 \c_hash_str \l__stex_notation_variant_str
2447                 \c_hash_str \l__stex_notation_lang_str _cs
2448             } } { \l_tmpa_tl } $

```

```

2449     }
2450   }
2451 }
2452 }
2453 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

2454 \keys_define:nn { stex / setnotation } {
2455   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2456   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2457   unknown .code:n = \str_set:Nx
2458     \l__stex_notation_variant_str \l_keys_key_str
2459 }
2460
2461 \cs_new_protected:Nn \_stex_setnotation_args:n {
2462   \str_clear:N \l__stex_notation_lang_str
2463   \str_clear:N \l__stex_notation_variant_str
2464   \keys_set:nn { stex / setnotation } { #1 }
2465 }
2466
2467 \NewDocumentCommand \setnotation {m m} {
2468   \stex_get_symbol:n { #1 }
2469   \_stex_setnotation_args:n { #2 }
2470   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2471     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2472     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2473       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2474     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2475       { \c_hash_str }
2476     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2477       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2478     \exp_args:Nx \stex_add_to_current_module:n {
2479       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2480         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2481       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2482         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2483       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2484         { \c_hash_str }
2485     }
2486     \stex_debug:nn {notations}{
2487       Setting~default~notation~
2488       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2489       \l_stex_get_symbol_uri_str \
2490       \expandafter\meaning\csname
2491         l_stex_symdecl \l_stex_get_symbol_uri_str _notations\endcsname
2492     }
2493   }{
2494     % todo throw error
2495   }
2496 }
2497

```

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

\symdef

```
2498 \keys_define:nn { stex / symdef } {
2499   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2500   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2501   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2502   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2503   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2504   op        .tl_set:N   = \l__stex_notation_op_tl ,
2505   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2506   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2507   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2508   unknown   .code:n     = \str_set:Nx
2509             \l__stex_notation_variant_str \l_keys_key_str
2510 }
2511
2512 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2513   \str_clear:N \l_stex_symdecl_name_str
2514   \str_clear:N \l_stex_symdecl_args_str
2515   \bool_set_false:N \l_stex_symdecl_local_bool
2516   \tl_clear:N \l_stex_symdecl_type_tl
2517   \tl_clear:N \l_stex_symdecl_definiens_tl
2518   \str_clear:N \l__stex_notation_lang_str
2519   \str_clear:N \l__stex_notation_variant_str
2520   \str_clear:N \l__stex_notation_prec_str
2521   \tl_clear:N \l__stex_notation_op_tl
2522
2523   \keys_set:nn { stex / symdef } { #1 }
2524 }
2525
2526 \NewDocumentCommand \symdef { 0{} m } {
2527   \__stex_notation_symdef_args:n { #1 }
2528   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2529   \stex_symdecl_do:n { #2 }
2530   \exp_args:Nx \stex_notation_do:nn {
2531     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2532   }
2533 }
2534 \stex_deactivate_macro:Nn \symdef {module~environments}
2535
2536 (End definition for \symdef. This function is documented on page 37.)
2537 \endpackage
```


Chapter 31

STEX -Terms Implementation

```
2536 <*package>
2537
2538 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2539
2540 <@@=stex_terms>
2541
2542   Warnings and error messages
2543   \msg_new:nnn{stex}{error/nonotation}{
2544     Symbol~#1~invoked,~but~has~no~notation~#2!
2545   }
2546   \msg_new:nnn{stex}{error/notationarg}{
2547     Error~in~parsing~notation~#1
2548   }
2549   \msg_new:nnn{stex}{error/noop}{
2550     Symbol~#1~has~no~operator~notation~for~notation~#2
2551   }
```

31.1 Symbol Invocations

Arguments:

```
2551 \keys_define:nn { stex / terms } {
2552   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2553   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2554   unknown .code:n = \str_set:Nx
2555     \l__stex_terms_variant_str \l_keys_key_str
2556 }
2557
2558 \cs_new_protected:Nn \__stex_terms_args:n {
2559   \str_clear:N \l__stex_terms_lang_str
2560   \str_clear:N \l__stex_terms_variant_str
2561   \str_clear:N \l__stex_terms_prec_str
2562   \tl_clear:N \l__stex_terms_op_tl
2563
2564   \keys_set:nn { stex / terms } { #1 }
```

2565 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2566 \cs_new_protected:Nn \stex_invoke_symbol:n {
2567   \if_mode_math:
2568     \exp_after:wN \__stex_terms_invoke_math:n
2569   \else:
2570     \exp_after:wN \__stex_terms_invoke_text:n
2571   \fi: { #1 }
2572 }
```

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

__stex_terms_invoke_math:n

```
2573 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2574   \peek_charcode_remove:NTF ! {
2575     \peek_charcode:NTF [ {
2576       \__stex_terms_invoke_op:nw { #1 }
2577     }{
2578       \peek_charcode_remove:NTF ! {
2579         \peek_charcode:NTF [ {
2580           \__stex_terms_invoke_op_custom:nw
2581         }{
2582           % TODO throw error
2583         }
2584       }{
2585         \__stex_terms_invoke_op:nw { #1 } []
2586       }
2587     }
2588   }{
2589     \peek_charcode_remove:NTF * {
2590       \__stex_terms_invoke_text:n { #1 }
2591     }{
2592       \peek_charcode:NTF [ {
2593         \__stex_terms_invoke_math:nw { #1 }
2594       }{
2595         \__stex_terms_invoke_math:nw { #1 } []
2596       }
2597     }
2598   }
2599 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2600 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2601   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2602     \stex_highlight_term:nn{#1}{#2}
2603   }
2604 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

__stex_terms_invoke_op:nw

```

2605 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2606   \__stex_terms_args:n { #2 }
2607   \cs_if_exist:cTF {
2608     stex_op_notation_ #1 \c_hash_str
2609     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2610   }{
2611     \csname stex_op_notation_ #1 \c_hash_str
2612       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2613     \endcsname
2614   }{
2615     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2616   }
2617 }

```

(End definition for __stex_terms_invoke_op:nw.)

__stex_terms_invoke_math:nw

```

2618 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2619   \__stex_terms_args:n { #2 }
2620   \seq_if_empty:cTF {
2621     l_stex_symdecl_ #1 _notations
2622   } {
2623     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2624   } {
2625     \seq_if_in:cxTF {
2626       l_stex_symdecl_ #1 _notations
2627     }
2628     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2629       \str_set:Nn \l_stex_current_symbol_str { #1 }
2630       \use:c{
2631         stex_notation_ #1 \c_hash_str
2632         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2633         _cs
2634       }
2635     }{
2636       \str_if_empty:NTF \l__stex_terms_variant_str {
2637         \str_if_empty:NTF \l__stex_terms_lang_str {
2638           \seq_get_left:cN {
2639             l_stex_symdecl_ #1 _notations
2640           } \l_tmpa_str
2641           \str_set:Nn \l_stex_current_symbol_str { #1 }
2642           \use:c{
2643             stex_notation_ #1 \c_hash_str \l_tmpa_str
2644             _cs
2645           }
2646         }{
2647           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2648             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2649           }
2650         }
2651       }{
2652         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2653           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2654     }
2655   }
2656 }
2657 }
2658 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2659 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2660   \peek_charcode_remove:NTF ! {
2661     \stex_term_custom:nn { #1 } { }
2662   }{
2663     \prop_set_eq:Nc \l_tmpa_prop {
2664       l_stex_symdecl_ #1 _prop
2665     }
2666     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2667     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2668   }
2669 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2670 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2671 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2672 \int_new:N \l__stex_terms_downprec
2673 \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
2674 \tl_set:Nn \l__stex_terms_left_bracket_str (
2675 \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

```

2676 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2677   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2678     \bool_set_false:N \l__stex_terms_brackets_done_bool
2679     #2
2680   } {
2681     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2682       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2683         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2684         \dobrackets { #2 }
2685       }

```

```

2686     }{ #2 }
2687   }
2688 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2689 \bool_new:N \l__stex_terms_brackets_done_bool
2690 %\RequirePackage{scalerel}
2691 \cs_new_protected:Npn \dobrackets #1 {
2692   %\ThisStyle{\if D\m@switch
2693   %   \exp_args:Nnx \use:nn
2694   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2695   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2696   % \else
2697   \exp_args:Nnx \use:nn
2698   {
2699     \bool_set_true:N \l__stex_terms_brackets_done_bool
2700     \int_set:Nn \l__stex_terms_downprec \infprec
2701     \l__stex_terms_left_bracket_str
2702     #1
2703   }
2704   {
2705     \bool_set_false:N \l__stex_terms_brackets_done_bool
2706     \l__stex_terms_right_bracket_str
2707     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2708   }
2709   %\fi}
2710 }

```

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

`\withbrackets`

```

2711 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2712   \exp_args:Nnx \use:nn
2713   {
2714     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2715     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2716     #3
2717   }
2718   {
2719     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2720     {\l__stex_terms_left_bracket_str}
2721     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2722     {\l__stex_terms_right_bracket_str}
2723   }
2724 }

```

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

`\STEXinvisible`

```

2725 \cs_new_protected:Npn \STEXinvisible #1 {
2726   \stex_annotate_invisible:n { #1 }
2727 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2728 \cs_new_protected:Nn \_stex_term_oms:nnn {
2729   \stex_annotate:nnn{ OMID }{ #2 }{
2730     \stex_highlight_term:nn { #1 } { #3 }
2731   }
2732 }
2733
2734 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2735   \__stex_terms_maybe_brackets:nn { #3 }{
2736     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2737   }
2738 }

```

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

`_stex_term_math_oma:nnnn`

```

2739 \cs_new_protected:Nn \_stex_term_oma:nnn {
2740   \stex_annotate:nnn{ OMA }{ #2 }{
2741     \stex_highlight_term:nn { #1 } { #3 }
2742   }
2743 }
2744
2745 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2746   \__stex_terms_maybe_brackets:nn { #3 }{
2747     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2748   }
2749 }

```

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

`_stex_term_math_omb:nnnn`

```

2750 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2751   \stex_annotate:nnn{ OMBIND }{ #2 }{
2752     \stex_highlight_term:nn { #1 } { #3 }
2753   }
2754 }
2755
2756 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2757   \__stex_terms_maybe_brackets:nn { #3 }{
2758     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2759   }
2760 }

```

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

`_stex_term_math_arg:nnn`

```

2761 \cs_new_protected:Nn \_stex_term_arg:nn {
2762   \stex_unhighlight_term:n {
2763     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2764   }
2765 }

```

```

2766 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2767   \exp_args:Nnx \use:nn
2768     { \int_set:Nn \l__stex_terms_downprec { #2 }
2769       \stex_term_arg:nn { #1 }{ #3 }
2770     }
2771   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2772 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2773 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2774   \clist_set:Nn \l_tmpa_clist{ #4 }
2775   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2776     \tl_set:Nn \l_tmpa_tl { #4 }
2777   }{
2778     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2779     \clist_reverse:N \l_tmpa_clist
2780     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2781
2782     \clist_map_inline:Nn \l_tmpa_clist {
2783       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2784         \exp_args:Nno
2785           \l_tmpa_cs { ##1 } \l_tmpa_tl
2786       }
2787     }
2788
2789   }
2790   \exp_args:Nnno
2791   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2792 }

```

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

`\stex_term_custom:nn`

```

2793 \cs_new_protected:Nn \stex_term_custom:nn {
2794   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2795   \str_set:Nn \l_tmpa_str { #2 }
2796   \tl_clear:N \l_tmpa_tl
2797   \int_zero:N \l_tmpa_int
2798   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2799   \__stex_terms_custom_loop:
2800 }

```

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

`__stex_terms_custom_loop:`

```

2801 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2802   \bool_set_false:N \l_tmpa_bool
2803   \bool_while_do:nn {
2804     \str_if_eq_p:ee X {
2805       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2806     }
2807   }{
2808     \int_incr:N \l_tmpa_int

```

```

2809 }
2810
2811 \peek_charcode:NTF [ {
2812   % notation/text component
2813   \__stex_terms_custom_component:w
2814 } {
2815   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2816     % all arguments read => finish
2817     \__stex_terms_custom_final:
2818   } {
2819     % arguments missing
2820     \peek_charcode_remove:NTF * {
2821       % invisible, specific argument position or both
2822       \peek_charcode:NTF [ {
2823         % visible specific argument position
2824         \__stex_terms_custom_arg:wn
2825       } {
2826         % invisible
2827         \peek_charcode_remove:NTF * {
2828           % invisible specific argument position
2829           \__stex_terms_custom_arg_inv:wn
2830         } {
2831           % invisible next argument
2832           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2833         }
2834       }
2835     } {
2836       % next normal argument
2837       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2838     }
2839   }
2840 }
2841 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2842 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2843   \bool_set_true:N \l_tmpa_bool
2844   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2845 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2846 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2847   \str_set:Nx \l_tmpb_str {
2848     \str_item:Nn \l_tmpa_str { #1 }
2849   }
2850   \str_case:VnTF \l_tmpb_str {
2851     { X } {
2852       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2853     }
2854     { i } { \__stex_terms_custom_set_X:n { #1 } }
2855     { b } { \__stex_terms_custom_set_X:n { #1 } }

```



```

2856     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2857     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2858   }{}{
2859     \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2860   }
2861
2862   \bool_if:nTF \l_tmpa_bool {
2863     \tl_put_right:Nx \l_tmpa_tl {
2864       \stex_annotate_invisible:n {
2865         \_stex_term_arg:nn { \int_eval:n { #1 } }
2866         \exp_not:n { { #2 } }
2867       }
2868     }
2869   } {
2870     \tl_put_right:Nx \l_tmpa_tl {
2871       \_stex_term_arg:nn { \int_eval:n { #1 } }
2872       \exp_not:n { { #2 } }
2873     }
2874   }
2875
2876   \_stex_terms_custom_loop:
2877 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2878 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2879   \str_set:Nx \l_tmpa_str {
2880     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2881     X
2882     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2883   }
2884 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2885 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2886   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2887   \_stex_terms_custom_loop:
2888 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2889 \cs_new_protected:Nn \_stex_terms_custom_final: {
2890   \int_compare:nNnTF \l_tmpb_int = 0 {
2891     \exp_args:Nnno \_stex_term_oms:nnn
2892   }{
2893     \str_if_in:NnTF \l_tmpa_str {b} {
2894       \exp_args:Nnno \_stex_term_ombind:nnn
2895     } {
2896       \exp_args:Nnno \_stex_term_oma:nnn
2897     }
2898   }

```

```

2899 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2900 }

```

(End definition for `_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2901 \NewDocumentCommand \symref { m m }{
2902   \let\compemph_uri_prev:\compemph@uri
2903   \let\compemph@uri\symrefemph@uri
2904   \STEXsymbol{#1}! [#2]
2905   \let\compemph@uri\compemph_uri_prev:
2906 }
2907
2908 \keys_define:nn { stex / symname } {
2909   post      .str_set_x:N    = \l_stex_symname_post_str
2910 }
2911
2912 \cs_new_protected:Nn \stex_symname_args:n {
2913   \str_clear:N \l_stex_symname_post_str
2914   \keys_set:nn { stex / symname } { #1 }
2915 }
2916
2917 \NewDocumentCommand \symname { O{} m }{
2918   \stex_symname_args:n { #1 }
2919   \stex_get_symbol:n { #2 }
2920   \str_set:Nx \l_tmpa_str {
2921     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2922   }
2923   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2924
2925   \let\compemph_uri_prev:\compemph@uri
2926   \let\compemph@uri\symrefemph@uri
2927   \exp_args:NNx \use:nn
2928   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2929     \l_tmpa_str \l_stex_symname_post_str
2930   ] }
2931   \let\compemph@uri\compemph_uri_prev:
2932 }

```

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

31.3 Notation Components

```

2933 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2934
2935 \str_new:N \l_stex_current_symbol_str
2936 \cs_new_protected:Nn \stex_highlight_term:nn {
2937   \exp_args:Nnx
2938   \use:nn {
2939     \str_set:Nx \l_stex_current_symbol_str { #1 }
2940     #2
2941   } {

```

```

2942 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2943 { \l_stex_current_symbol_str }
2944 }
2945 }
2946
2947 \cs_new_protected:Nn \stex_unhighlight_term:n {
2948 % \latexml_if:TF {
2949 % #1
2950 % } {
2951 % \rustex_if:TF {
2952 % #1
2953 % } {
2954 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2955 % }
2956 % }
2957 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2958 \cs_new_protected:Npn \comp #1 {
2959 \str_if_empty:NF \l_stex_current_symbol_str {
2960 \rustex_if:TF {
2961 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
2962 }{
2963 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2964 }
2965 }
2966 }
2967
2968 \cs_new_protected:Npn \compemph@uri #1 #2 {
2969 \compemph{ #1 }
2970 }
2971
2972
2973 \cs_new_protected:Npn \compemph #1 {
2974 #1
2975 }
2976
2977 \cs_new_protected:Npn \defemph@uri #1 #2 {
2978 \defemph{#1}
2979 }
2980
2981 \cs_new_protected:Npn \defemph #1 {
2982 \textbf{#1}
2983 }
2984
2985 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2986 \symrefemph{#1}
2987 }
2988
2989 \cs_new_protected:Npn \symrefemph #1 {
2990 \textbf{#1}
2991 }

```

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

`\ellipses`

```
2992 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2993 \bool_new:N \l_stex_inarray_bool
\parrayline 2994 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2995 \NewDocumentCommand \parray { m m } {
\parraycell 2996 \begin{group}
2997 \bool_set_true:N \l_stex_inarray_bool
2998 \begin{array}{#1}
2999 #2
3000 \end{array}
3001 \end{group}
3002 }
3003
3004 \NewDocumentCommand \prmatrix { m } {
3005 \begin{group}
3006 \bool_set_true:N \l_stex_inarray_bool
3007 \begin{matrix}
3008 #1
3009 \end{matrix}
3010 \end{group}
3011 }
3012
3013 \def \maybepline {
3014 \bool_if:NT \l_stex_inarray_bool {\hline}
3015 }
3016
3017 \def \parrayline #1 #2 {
3018 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3019 }
3020
3021 \def \pmrow #1 { \parrayline{}{ #1 } }
3022
3023 \def \parraylineh #1 #2 {
3024 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3025 }
3026
3027 \def \parraycell #1 {
3028 #1 \bool_if:NT \l_stex_inarray_bool {&}
3029 }

```

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

```
3030 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
3031 <*package>
3032
3033 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3034
3035 <@@=stex_features>
    Warnings and error messages
3036
```

32.1 Imports with modification

```
3037 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3038   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3039     \__stex_features_get_symbol_from_cs:n { #1 }
3040   }{
3041     % argument is a string
3042     % is it a command name?
3043     \cs_if_exist:cTF { #1 }{
3044       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3045       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3046       \str_if_empty:NNTF \l_tmpa_str {
3047         \exp_args:Nx \cs_if_eq:NNTF {
3048           \tl_head:N \l_tmpa_tl
3049         } \stex_invoke_symbol:n {
3050           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3051         }{
3052           \__stex_features_get_symbol_from_string:n { #1 }
3053         }
3054       } {
3055         \__stex_features_get_symbol_from_string:n { #1 }
3056       }
3057     }{
3058       % argument is not a command name
```

```

3059     \_stex_features_get_symbol_from_string:n { #1 }
3060     % \l_stex_all_symbols_seq
3061   }
3062 }
3063 }
3064
3065 \cs_new_protected:Nn \_stex_features_get_symbol_from_string:n {
3066   \str_set:Nn \l_tmpa_str { #1 }
3067   \bool_set_false:N \l_tmpa_bool
3068   \bool_if:NF \l_tmpa_bool {
3069     \tl_set:Nn \l_tmpa_tl {
3070       \msg_set:nnn{stex}{error/unknownsymbol}{
3071         No~symbol~#1~found!
3072       }
3073       \msg_error:nn{stex}{error/unknownsymbol}
3074     }
3075     \str_set:Nn \l_tmpa_str { #1 }
3076     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3077     \seq_map_inline:Nn \_stex_features_copymodule_fields_seq {
3078       \str_set:Nn \l_tmpb_str { ##1 }
3079       \str_if_eq:eeT { \l_tmpa_str } {
3080         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3081       } {
3082         \seq_map_break:n {
3083           \tl_set:Nn \l_tmpa_tl {
3084             \str_set:Nn \l_stex_get_symbol_uri_str {
3085               ##1
3086             }
3087             \_stex_features_get_symbol_check:
3088           }
3089         }
3090       }
3091     }
3092     \l_tmpa_tl
3093   }
3094 }
3095
3096 \cs_new_protected:Nn \_stex_features_get_symbol_from_cs:n {
3097   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3098   { \tl_tail:N \l_tmpa_tl }
3099   \tl_if_single:NTF \l_tmpa_tl {
3100     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3101       \exp_after:wN \str_set:Nn \exp_after:wN
3102       \l_stex_get_symbol_uri_str \l_tmpa_tl
3103       \_stex_features_get_symbol_check:
3104     }{
3105       % TODO
3106       % tail is not a single group
3107     }
3108   }{
3109     % TODO
3110     % tail is not a single group
3111   }
3112 }

```

```

3113
3114 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3115   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3116   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3117     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3118     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3119     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq {
3120       % TODO error
3121     }
3122   }{
3123     % TODO error
3124   }
3125 }
3126
3127 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3128   \stex_import_module_uri:nn { #1 } { #2 }
3129   \stex_deactivate_macro:Nn \symdecl {module~environments}
3130   \stex_deactivate_macro:Nn \symdef {module~environments}
3131   \stex_deactivate_macro:Nn \notation {module~environments}
3132   \stex_reactivate_macro:N \assign
3133   \stex_reactivate_macro:N \renamedec1
3134   \stex_reactivate_macro:N \donotcopy
3135   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3136   %\let\notation\notation_in_copymodules:
3137   %\stex_module_setup:nn {}{ #3 }
3138   \stex_import_require_module:nnnn
3139   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3140   { \l_stex_import_path_str } { \l_stex_import_name_str }
3141   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3142   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3143   \seq_clear:N \l__stex_features_copymodule_fields_seq
3144   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3145     \seq_map_inline:cn {c_stex_module_###1_constants}{
3146       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3147         ##1 ? #####1
3148       }
3149     }
3150   }
3151   \seq_clear:N \l_tmpa_seq
3152   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3153     name      = \l_stex_current_copymodule_name_str ,
3154     module    = \l_stex_current_module_str ,
3155     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3156     includes  = \l_tmpa_seq ,
3157     fields    = \l_tmpa_seq
3158   }
3159   \stex_debug:nn{copymodule}{cloning~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3160     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3161   \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3162   % todo
3163
3164   \stex_if_smsmode:TF {
3165     \stex_smsmode_set_codes:
3166   } {

```

```

3167 \begin{stex_annotate_env} {structure} {
3168 \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3169 }
3170 \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3171 }
3172 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3173 \bool_set_false:N \l_stex_html_do_output_bool
3174 }{
3175 \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3176 \tl_clear:N \l_tmpa_tl
3177 \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3178 \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3179 \seq_map_inline:cn {c_stex_module_###1_constants}{\stex_annotate:nnn{assignment} {###1?###1
3180 \str_if_exist:cTF {l__stex_features_copymodule_###1?###1_name_str} {
3181 \tl_put_right:Nx \l_tmpa_tl {
3182 \prop_set_from_keyval:cn {
3183 l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3184 }{
3185 \exp_after:wN \prop_to_keyval:N \csname
3186 l_stex_symdecl\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3187 \endcsname
3188 }
3189 \seq_clear:c {
3190 l_stex_symdecl_
3191 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3192 _notations
3193 }
3194 }
3195 \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_###1?###1_name_str}
3196 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3197 \str_if_exist:cT {l__stex_features_copymodule_###1?###1_macroname_str} {
3198 \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_###1?###1_name_str}
3199 \tl_put_right:Nx \l_tmpa_tl {
3200 \tl_set:cx {\use:c{l__stex_features_copymodule_###1?###1_macroname_str}}{
3201 \stex_invoke_symbol:n {
3202 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_###1?###1_name_str}
3203 }
3204 }
3205 }
3206 }
3207 }{
3208 \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ###1?###1 _prop}
3209 \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ###1 }
3210 \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3211 \tl_put_right:Nx \l_tmpa_tl {
3212 \prop_set_from_keyval:cn {
3213 l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3214 }{
3215 \prop_to_keyval:N \l_tmpa_prop
3216 }
3217 \seq_clear:c {
3218 l_stex_symdecl_
3219 \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ###1
3220 _notations

```



```

3221     }
3222   }
3223   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule
3224   \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3225     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3226     \tl_put_right:Nx \l_tmpa_tl {
3227       \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3228       \stex_invoke_symbol:n {
3229         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3230       }
3231     }
3232   }
3233 }
3234 }
3235 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3236   \stex_annotate_invisible:nnn{definiens}{\{$\use:c{l__stex_features_copymodule_##1?###
3237 }
3238 % todo notations
3239 }}
3240 }
3241 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3242 \tl_put_left:Nx \l_tmpa_tl {
3243   \prop_set_from_keyval:cn {
3244     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3245   }{
3246     \prop_to_keyval:N \l_stex_current_copymodule_prop
3247   }
3248 }
3249 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3250 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3251 \exp_args:Nx \stex_do_aftergroup:n {
3252   \exp_args:No \exp_not:n \l_tmpa_tl
3253 }
3254 \stex_if_smsmode:F {
3255   \end{stex_annotate_env}
3256 }
3257 }
3258
3259 \NewDocumentCommand \donotcopy { 0{} m}{
3260   \stex_import_module_uri:nn { #1 } { #2 }
3261   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3262   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3263     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3264     \seq_map_inline:cn {c_stex_module_##1_constants}{
3265       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? ####1 }
3266       \bool_lazy_any_p:nT {
3267         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3268         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3269         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3270       }{
3271         % TODO throw error
3272       }
3273     }
3274   }

```

```

3275
3276 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3277 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3278 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3279 }
3280
3281 \NewDocumentCommand \assign { m m }{
3282 \stex_get_symbol_in_copymodule:n {#1}
3283 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3284 \tl_set:cn {\l_stex_features_copymodule_##1?####1_def_tl}{#2}
3285 }
3286
3287 \keys_define:nn { stex / renamedec1 } {
3288 name .str_set_x:N = \l_stex_renamedec1_name_str
3289 }
3290 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3291 \str_clear:N \l_stex_renamedec1_name_str
3292
3293 \keys_set:nn { stex / renamedec1 } { #1 }
3294 }
3295
3296 \NewDocumentCommand \renamedec1 { 0{} m m }{
3297 \__stex_features_renamedec1_args:n { #1 }
3298 \stex_get_symbol_in_copymodule:n {#2}
3299 \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3300 \str_set:cx {\l_stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3301 \str_if_empty:NTF \l_stex_renamedec1_name_str {
3302 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3303 \l_stex_get_symbol_uri_str
3304 } }
3305 } {
3306 \str_set:cx {\l_stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3307 \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3308 \prop_set_eq:cc {\l_stex_symdecl_
3309 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3310 _prop
3311 }{\l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3312 \seq_set_eq:cc {\l_stex_symdecl_
3313 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3314 _notations
3315 }{\l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3316 \prop_put:cnx {\l_stex_symdecl_
3317 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3318 _prop
3319 }{ name }{ \l_stex_renamedec1_name_str }
3320 \prop_put:cnx {\l_stex_symdecl_
3321 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3322 _prop
3323 }{ module }{ \l_stex_current_module_str }
3324 \exp_args:NNx \seq_put_left:Nn \l_stex_features_copymodule_fields_seq {
3325 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3326 }
3327 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3328 \l_stex_current_module_str ? \l_stex_renamedec1_name_str

```

```

3329     } }
3330   }
3331 }
3332 %\NewDocumentCommand \notation_in_copymodules: { O{} m } {
3333 % \stex_notation_args:n { #1 }
3334 % \tl_clear:N \l_stex_symdecl_definiens_tl
3335 % \stex_get_symbol_in_copymodule:n { #2 }
3336 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3337 % % todo
3338 %}
3339 \stex_deactivate_macro:Nn \assign {copymodules}
3340 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3341 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3342
3343
3344 \seq_new:N \l_stex_implicit_morphisms_seq
3345 \NewDocumentCommand \implicitmorphism { O{} m m }{
3346   \stex_import_module_uri:nn { #1 } { #2 }
3347   \stex_debug:nn{implicits}{
3348     Implicit~morphism:~
3349     \l_stex_module_ns_str ? \l__stex_features_name_str
3350   }
3351   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3352     \l_stex_module_ns_str ? \l__stex_features_name_str
3353   }{
3354     \msg_error:nnn{stex}{error/conflictingmodules}{
3355       \l_stex_module_ns_str ? \l__stex_features_name_str
3356     }
3357   }
3358
3359   % TODO
3360
3361
3362
3363   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3364     \l_stex_module_ns_str ? \l__stex_features_name_str
3365   }
3366 }
3367

```

32.2 The feature environment

structural@feature

```

3368
3369 \NewDocumentEnvironment{structural@feature}{ m m m }{
3370   \stex_if_in_module:F {
3371     \msg_set:nnn{stex}{error/nomodule}{
3372       Structural~Feature~has~to~occur~in~a~module:\\
3373       Feature~#2~of~type~#1\\
3374       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3375     }
3376     \msg_error:nn{stex}{error/nomodule}
3377   }

```

```

3378
3379 \str_set:Nx \l_stex_module_name_str {
3380   \prop_item:Nn \l_stex_current_module_prop
3381     { name } / #2 - feature
3382 }
3383
3384 \str_set:Nx \l_stex_module_ns_str {
3385   \prop_item:Nn \l_stex_current_module_prop
3386     { ns }
3387 }
3388
3389
3390 \str_clear:N \l_tmpa_str
3391 \seq_clear:N \l_tmpa_seq
3392 \tl_clear:N \l_tmpa_tl
3393 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3394   origname = #2,
3395   name     = \l_stex_module_name_str ,
3396   ns       = \l_stex_module_ns_str ,
3397   imports  = \exp_not:o { \l_tmpa_seq } ,
3398   constants = \exp_not:o { \l_tmpa_seq } ,
3399   content  = \exp_not:o { \l_tmpa_tl } ,
3400   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3401   lang     = \l_stex_module_lang_str ,
3402   sig      = \l_tmpa_str ,
3403   meta     = \l_tmpa_str ,
3404   feature  = #1 ,
3405 }
3406
3407 \stex_if_smsmode:TF {
3408   \stex_smsmode_set_codes:
3409 } {
3410   \begin{stex_annotate_env}{ feature:#1 }{}
3411   \stex_annotate_invisible:nnn{header}{}{ #3 }
3412 }
3413 }{
3414   \str_set:Nx \l_tmpa_str {
3415     c_stex_feature_
3416     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3417     \prop_item:Nn \l_stex_current_module_prop { name }
3418     _prop
3419   }
3420   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3421   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3422   \stex_if_smsmode:TF {
3423     \exp_args:Nx \stex_add_to_sms:n {
3424       \prop_gset_from_keyval:cn {
3425         c_stex_feature_
3426         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3427         \prop_item:Nn \l_stex_current_module_prop { name }
3428         _prop
3429       } {
3430         origname = #2,
3431         name     = \prop_item:cn { \l_tmpa_str } { name } ,

```

```

3432         ns      = \prop_item:cn { \l_tmpa_str } { ns } ,
3433         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3434         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3435         content  = \prop_item:cn { \l_tmpa_str } { content } ,
3436         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3437         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3438         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3439         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3440         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3441     }
3442 }
3443 } {
3444     \end{stex_annotate_env}
3445 }
3446 }
3447

```

32.3 Features

structure

```

3448
3449 \prop_new:N \l_stex_all_structures_prop
3450
3451 \keys_define:nn { stex / features / structure } {
3452     name          .str_set:x:N = \l__stex_features_structure_name_str ,
3453 }
3454
3455 \cs_new_protected:Nn \__stex_features_structure_args:n {
3456     \str_clear:N \l__stex_features_structure_name_str
3457     \keys_set:nn { stex / features / structure } { #1 }
3458 }
3459
3460 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3461 %   \__stex_features_structure_args:n { ##1 }
3462 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3463 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3464 %   }
3465 %} {
3466 %
3467 %}
3468
3469 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3470     \__stex_features_structure_args:n { #1 }
3471     \str_if_empty:NT \l__stex_features_structure_name_str {
3472         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3473     }
3474     \exp_args:Nnnx
3475     \begin{structural@feature}{ structure }
3476         { \l__stex_features_structure_name_str }{}
3477         \seq_clear:N \l_tmpa_seq
3478         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3479     }
3480 }{

```

```

3481 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3482 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3483 \str_set:Nx \l_tmpa_str {
3484   \prop_item:Nn \l_stex_current_module_prop { ns } ?
3485   \prop_item:Nn \l_stex_current_module_prop { name }
3486 }
3487 \seq_map_inline:Nn \l_tmpa_seq {
3488   \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3489 }
3490 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3491 \exp_args:Nnx
3492 \AddToHookNext { env / mathstructure / after } {
3493   \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3494     \stex_term_math_oms:nnnn { \l_tmpa_str }{}{}{}
3495   }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3496 \STEXexport {
3497   \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3498     {\prop_item:Nn \l_stex_current_module_prop { origname }}
3499     {\l_tmpa_str}
3500   \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3501     {#2}{\l_tmpa_str}
3502   % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3503   %   \prop_item:Nn \l_stex_current_module_prop { origname },
3504   %   \l_tmpa_str
3505   % }
3506   % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3507   %   #2,\l_tmpa_str
3508   % }
3509   % \tl_set:cx { #2 } {
3510   %   \stex_invoke_structure:n { \l_tmpa_str }
3511   % }
3512 }
3513
3514 \end{structural@feature}
3515 % \g_stex_last_feature_prop
3516 }

```

\instantiate

```

3517 \seq_new:N \l__stex_features_structure_field_seq
3518 \str_new:N \l__stex_features_structure_field_str
3519 \str_new:N \l__stex_features_structure_def_tl
3520 \prop_new:N \l__stex_features_structure_prop
3521 \NewDocumentCommand \instantiate { m O{} m }{
3522   \stex_smsmode_set_codes:
3523   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3524   \prop_set_eq:Nc \l__stex_features_structure_prop {
3525     c_stex_feature_\l_tmpa_str _prop
3526   }
3527   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3528   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3529     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3530     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3531       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3532       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq

```

```

3533     {!} \l_tmpa_tl
3534 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3535     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3536     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3537     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3538 }{
3539     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3540     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3541     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3542     \l_tmpa_tl
3543     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3544         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3545         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3546     }{
3547         \tl_clear:N \l_tmpb_tl
3548     }
3549 }
3550 }{
3551     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3552     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3553         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3554         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3555         \tl_clear:N \l_tmpa_tl
3556     }{
3557         % TODO throw error
3558     }
3559 }
3560 % \l_tmpa_str: name
3561 % \l_tmpa_tl: definiens
3562 % \l_tmpb_tl: notation
3563 \tl_if_empty:NT \l__stex_features_structure_field_str {
3564     % TODO throw error
3565 }
3566 \str_clear:N \l_tmpb_str
3567
3568 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3569 \seq_map_inline:Nn \l_tmpa_seq {
3570     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3571     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3572     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3573         \seq_map_break:n {
3574             \str_set:Nn \l_tmpb_str { ####1 }
3575         }
3576     }
3577 }
3578 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3579     \l_tmpb_str
3580
3581 \tl_if_empty:NTF \l_tmpb_tl {
3582     \tl_if_empty:NF \l_tmpa_tl {
3583         \exp_args:Nx \use:n {
3584             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3585         }
3586     }

```

```

3587   }{
3588     \tl_if_empty:NTF \l_tmpa_tl {
3589       \exp_args:Nx \use:n {
3590         \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3591       }
3592
3593     }{
3594       \exp_args:Nx \use:n {
3595         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3596         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3597       }
3598     }
3599   }
3600   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3601   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3602   % #3/\l__stex_features_structure_field_str
3603   % \par
3604   % \expandafter\present\csname
3605   %   l_stex_symdecl_
3606   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3607   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3608   %   #3/\l__stex_features_structure_field_str
3609   %   _prop
3610   % \endcsname
3611 }
3612
3613 \tl_clear:N \l__stex_features_structure_def_tl
3614
3615 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3616 \seq_map_inline:Nn \l_tmpa_seq {
3617   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3618   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3619   \exp_args:Nx \use:n {
3620     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3621
3622     }
3623   }
3624
3625   \prop_if_exist:cF {
3626     l_stex_symdecl_
3627     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3628     \prop_item:Nn \l_stex_current_module_prop {name} ?
3629     #3/\l_tmpa_str
3630     _prop
3631   }{
3632     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3633     \l_tmpb_str
3634     \exp_args:Nx \use:n {
3635       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3636     }
3637   }
3638 }
3639
3640 \symdecl*[type={\STEXsymbol{module-type}}{

```



```

3641 \stex_term_math_oms:nnnn {
3642   \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3643   \prop_item:Nn \l__stex_features_structure_prop {name}
3644   }{}{0}{}
3645 }{}{#3}
3646
3647 % TODO: -> sms file
3648
3649 \tl_set:cx{ #3 }{
3650   \stex_invoke_structure:nnn {
3651     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3652     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3653   } {
3654     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3655     \prop_item:Nn \l__stex_features_structure_prop {name}
3656   }
3657 }
3658
3659 }

```

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

\stex_invoke_structure:nnn

```

3660 % #1: URI of the instance
3661 % #2: URI of the instantiated module
3662 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3663   \tl_if_empty:nTF{ #3 }{
3664     \prop_set_eq:Nc \l__stex_features_structure_prop {
3665       c_stex_feature_ #2 _prop
3666     }
3667     \tl_clear:N \l_tmpa_tl
3668     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3669     \seq_map_inline:Nn \l_tmpa_seq {
3670       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3671       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3672       \cs_if_exist:cT {
3673         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3674       }{
3675         \tl_if_empty:NF \l_tmpa_tl {
3676           \tl_put_right:Nn \l_tmpa_tl {,}
3677         }
3678         \tl_put_right:Nx \l_tmpa_tl {
3679           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3680         }
3681       }
3682     }
3683     \exp_args:No \mathstruct \l_tmpa_tl
3684   }{
3685     \stex_invoke_symbol:n{#1/#3}
3686   }
3687 }

```

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

3688 </package>

Chapter 33

STEX -Statements Implementation

```
3689 <*package>
3690
3691 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3692
3693 \protected\def\ignorespacesandpars{
3694   \begingroup\catcode13=10\relax
3695   \@ifnextchar\par{
3696     \endgroup\expandafter\ignorespacesandpars\@gobble
3697   }{
3698     \endgroup
3699   }
3700 }
3701
3702 <@@=stex_statements>
3703
3704   Warnings and error messages
```

\titleemph

```
3704 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3705 \keys_define:nn {stex / definiendum }{
3706   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3707   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3708   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3709 }
3710 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3711   \str_clear:N \l__stex_statements_definiendum_root_str
3712   \tl_clear:N \l__stex_statements_definiendum_post_tl
3713   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3714 \keys_set:nn { stex / definiendum } { #1 }
3715 }
3716 \NewDocumentCommand \definiendum { 0{} m m } {
3717   \__stex_statements_definiendum_args:n { #1 }
3718   \stex_get_symbol:n { #2 }
3719   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3720   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3721     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3722       \tl_set:Nn \l_tmpa_tl { #3 }
3723     } {
3724       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3725       \tl_set:Nn \l_tmpa_tl {
3726         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3727       }
3728     }
3729   } {
3730     \tl_set:Nn \l_tmpa_tl { #3 }
3731   }
3732
3733   % TODO root
3734   \rustex_if:TF {
3735     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3736   } {
3737     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3738   }
3739 }
3740 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3741 \NewDocumentCommand \definame { 0{} m } {
3742   \__stex_statements_definiendum_args:n { #1 }
3743   % TODO: root
3744   \stex_get_symbol:n { #2 }
3745   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3746   \str_set:Nx \l_tmpa_str {
3747     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3748   }
3749   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3750   \rustex_if:TF {
3751     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3752       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3753     }
3754   } {
3755     \defemph@uri {
3756       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3757     } { \l_stex_get_symbol_uri_str }
3758   }
3759 }
3760 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3761
3762 \keys_define:nn {stex / sdefinition }{
3763   type      .str_set_x:N = \sdefinitiontype,
3764   id        .str_set_x:N = \sdefinitionid,
3765   title     .tl_set:N    = \sdefinitiontitle
3766 }
3767 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3768   \str_clear:N \sdefinitiontype
3769   \str_clear:N \sdefinitionid
3770   \tl_clear:N \sdefinitiontitle
3771   \keys_set:nn { stex / sdefinition }{ #1 }
3772 }
3773
3774 \NewDocumentEnvironment{sdefinition}{0{}}{
3775   \__stex_statements_sdefinition_args:n{ #1 }
3776   \stex_reactivate_macro:N \definiendum
3777   \stex_reactivate_macro:N \definame
3778   \stex_smsmode_set_codes:
3779   \clist_set:No \l_tmpa_clist \sdefinitiontype
3780   \tl_clear:N \l_tmpa_tl
3781   \clist_map_inline:Nn \l_tmpa_clist {
3782     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3783       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3784     }
3785   }
3786   \stex_if_smsmode:F {
3787     \exp_args:Nnnx
3788     \begin{stex_annotate_env}{definition}{}
3789     \str_if_empty:NF \sdefinitiontype {
3790       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3791     }
3792   }
3793   \tl_if_empty:NTF \l_tmpa_tl {
3794     \__stex_statements_sdefinition_start:
3795   }{
3796     \l_tmpa_tl
3797   }
3798   \stex_ref_new_doc_target:n \sdefinitionid
3799 }{
3800   \clist_set:No \l_tmpa_clist \sdefinitiontype
3801   \tl_clear:N \l_tmpa_tl
3802   \clist_map_inline:Nn \l_tmpa_clist {
3803     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3804       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3805     }
3806   }
3807   \tl_if_empty:NTF \l_tmpa_tl {
3808     \__stex_statements_sdefinition_end:
3809   }{
3810     \l_tmpa_tl
3811   }
3812   \stex_if_smsmode:F {
3813     \end{stex_annotate_env}

```

```

3814 }
3815 }

```

`\stexpatchdefinition`

```

3816 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3817   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3818     ~(\sdefinitiontitle)
3819   }~}
3820 }
3821 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3822
3823 \newcommand\stexpatchdefinition[3] [] {
3824   \str_set:Nx \l_tmpa_str{ #1 }
3825   \str_if_empty:NTF \l_tmpa_str {
3826     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3827     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3828   }{
3829     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3830     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3831   }
3832 }

```

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

`\inlinedef inline:`

```

3833 \NewDocumentCommand \inlinedef { m } {
3834   \begingroup
3835   \stex_reactivate_macro:N \definiendum
3836   \stex_reactivate_macro:N \definame
3837   \stex_ref_new_doc_target:n{
3838     #1
3839   }
3840 }

```

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

33.2 Assertions

`sassertion`

```

3841
3842 \keys_define:nn {stex / sassertion }{
3843   type      .str_set_x:N = \sassertiontype,
3844   id        .str_set_x:N = \sassertionid,
3845   title     .tl_set:N     = \sassertiontitle ,
3846   name      .str_set_x:N = \sassertionname
3847 }
3848 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3849   \str_clear:N \sassertiontype
3850   \str_clear:N \sassertionid
3851   \str_clear:N \sassertionname
3852   \tl_clear:N \sassertiontitle
3853   \keys_set:nn { stex / sassertion }{ #1 }
3854 }

```

```

3855
3856 %\tl_new:N \g__stex_statements_aftergroup_tl
3857
3858 \NewDocumentEnvironment{sassertion}{0{}}{
3859   \__stex_statements_sassertion_args:n{ #1 }
3860   \stex_smsmode_set_codes:
3861   \clist_set:No \l_tmpa_clist \sassertiontype
3862   \tl_clear:N \l_tmpa_tl
3863   \clist_map_inline:Nn \l_tmpa_clist {
3864     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3865       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3866     }
3867   }
3868   \stex_if_smsmode:F {
3869     \exp_args:Nnnx
3870     \begin{stex_annotate_env}{assertion}{}
3871     \str_if_empty:NF \sassertiontype {
3872       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3873     }
3874   }
3875   \tl_if_empty:NTF \l_tmpa_tl {
3876     \__stex_statements_sassertion_start:
3877   }{
3878     \l_tmpa_tl
3879   }
3880   \stex_ref_new_doc_target:n \sassertionid
3881 }{
3882   \clist_set:No \l_tmpa_clist \sassertiontype
3883   \tl_clear:N \l_tmpa_tl
3884   \clist_map_inline:Nn \l_tmpa_clist {
3885     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3886       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3887     }
3888   }
3889   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
3890   \tl_if_empty:NTF \l_tmpa_tl {
3891     \__stex_statements_sassertion_end:
3892   }{
3893     \l_tmpa_tl
3894   }
3895   \stex_if_smsmode:F {
3896     \end{stex_annotate_env}
3897   }
3898 }

```

\stexpatchassertion

```

3899
3900 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3901   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
3902     (\sassertiontitle)
3903   }~}
3904 }
3905 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3906

```

```

3907 \newcommand\stexpatchassertion[3] [] {
3908   \str_set:Nx \l_tmpa_str{ #1 }
3909   \str_if_empty:NTF \l_tmpa_str {
3910     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3911     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3912   }{
3913     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3914     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3915   }
3916 }

```

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

\inlineass inline:

```

3917 \NewDocumentCommand \inlineass { m } {
3918   \begingroup
3919   \stex_ref_new_doc_target:n{
3920     #1
3921   \endgroup
3922 }

```

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

33.3 Examples

sexample

```

3923
3924 \keys_define:nn {stex / sexample }{
3925   type      .str_set_x:N = \exampletype,
3926   id        .str_set_x:N = \sexampleid,
3927   title     .tl_set:N = \sexampletitle,
3928   for       .clist_set:N = \sexamplefor,
3929 }
3930 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3931   \str_clear:N \sexampletype
3932   \str_clear:N \sexampleid
3933   \tl_clear:N \sexampletitle
3934   \clist_clear:N \sexamplefor
3935   \keys_set:nn { stex / sexample }{ #1 }
3936 }
3937
3938 \NewDocumentEnvironment{sexample}{0{}}{
3939   \__stex_statements_sexample_args:n{ #1 }
3940   \stex_smsmode_set_codes:
3941   \clist_set:N \l_tmpa_clist \sexampletype
3942   \tl_clear:N \l_tmpa_tl
3943   \clist_map_inline:Nn \l_tmpa_clist {
3944     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
3945       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3946     }
3947   }
3948   \stex_if_smsmode:F {
3949     \seq_clear:N \l_tmpa_seq

```

```

3950 \clist_map_inline:Nn \sexamplefor {
3951 \str_if_eq:nnF{ ##1 }{}{
3952 \stex_get_symbol:n { ##1 }
3953 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3954 \l_stex_get_symbol_uri_str
3955 }
3956 }
3957 }
3958 \exp_args:Nnnx
3959 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3960 \str_if_empty:NF \sexamplotype {
3961 \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
3962 }
3963 }
3964 \tl_if_empty:NTF \l_tmpa_tl {
3965 \__stex_statements_sexample_start:
3966 }{
3967 \l_tmpa_tl
3968 }
3969 \stex_ref_new_doc_target:n \sexampleid
3970 }{
3971 \clist_set:N \l_tmpa_clist \sexamplotype
3972 \tl_clear:N \l_tmpa_tl
3973 \clist_map_inline:Nn \l_tmpa_clist {
3974 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3975 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3976 }
3977 }
3978 \tl_if_empty:NTF \l_tmpa_tl {
3979 \__stex_statements_sexample_end:
3980 }{
3981 \l_tmpa_tl
3982 }
3983 \stex_if_smsmode:F {
3984 \end{stex_annotate_env}
3985 }
3986 }

```

\stexpatchexample

```

3987
3988 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3989 \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplotype {
3990 (\sexamplotype)
3991 }~}
3992 }
3993 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3994
3995 \newcommand\stexpatchexample[3]{} {
3996 \str_set:Nx \l_tmpa_str{ #1 }
3997 \str_if_empty:NTF \l_tmpa_str {
3998 \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3999 \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4000 }{
4001 \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }

```



```

4002     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4003   }
4004 }

```

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

`\inlineex` inline:

```

4005 \NewDocumentCommand \inlineex { m } {
4006   \begingroup
4007   \stex_ref_new_doc_target:n{
4008     #1
4009   \endgroup
4010 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4011 \keys_define:nn { stex / sparagraph } {
4012   id      .str_set:x:N = \sparagraphid ,
4013   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4014   type    .str_set:x:N = \sparagraphtype ,
4015   for     .str_set:x:N = \sparagraphfor ,
4016   from    .tl_set:x:N  = \sparagraphfrom ,
4017   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4018   name    .str_set:N    = \sparagraphname
4019 }
4020
4021 \cs_new_protected:Nn \stex_sparagraph_args:n {
4022   \tl_clear:N \l_stex_sparagraph_title_tl
4023   \tl_clear:N \sparagraphfrom
4024   \tl_clear:N \l_stex_sparagraph_start_tl
4025   \str_clear:N \sparagraphid
4026   \str_clear:N \sparagraphtype
4027   \str_clear:N \sparagraphfor
4028   \str_clear:N \sparagraphname
4029   \keys_set:nn { stex / sparagraph }{ #1 }
4030 }
4031 \newif\if@in@omtext\@in@omtextfalse
4032
4033 \NewDocumentEnvironment {sparagraph} { 0{} } {
4034   \stex_sparagraph_args:n { #1 }
4035   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4036     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4037   }{
4038     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4039   }
4040   \@in@omtexttrue
4041   \stex_smsmode_set_codes:
4042   \clist_set:No \l_tmpa_clist \sparagraphtype
4043   \tl_clear:N \l_tmpa_tl
4044   \clist_map_inline:Nn \l_tmpa_clist {

```

```

4045 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4046 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4047 }
4048 }
4049 \stex_if_smsmode:F {
4050 \exp_args:Nnnx
4051 \begin{stex_annotate_env}{paragraph}{}
4052 \str_if_empty:NF \sparagraphtype {
4053 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4054 }
4055 }
4056 \tl_if_empty:NTF \l_tmpa_tl {
4057 \__stex_statements_sparagraph_start:
4058 }{
4059 \l_tmpa_tl
4060 }
4061 \stex_ref_new_doc_target:n \sparagraphid
4062 \ignorespacesandpars
4063 }{
4064 \clist_set:No \l_tmpa_clist \sparagraphtype
4065 \tl_clear:N \l_tmpa_tl
4066 \clist_map_inline:Nn \l_tmpa_clist {
4067 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4068 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4069 }
4070 }
4071 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4072 \tl_if_empty:NTF \l_tmpa_tl {
4073 \__stex_statements_sparagraph_end:
4074 }{
4075 \l_tmpa_tl
4076 }
4077 \stex_if_smsmode:F {
4078 \end{stex_annotate_env}
4079 }
4080 }

```

\stexpatchparagraph

```

4081
4082 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4083 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4084 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4085 \titleemph{\l_stex_sparagraph_title_tl}:~
4086 }
4087 }{
4088 \titleemph{\l_stex_sparagraph_start_tl}~
4089 }
4090 }
4091 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4092
4093 \newcommand\stexpatchparagraph[3]{} {
4094 \str_set:Nx \l_tmpa_str{ #1 }
4095 \str_if_empty:NTF \l_tmpa_str {
4096 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }

```

```

4097     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4098   }{
4099     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4100     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4101   }
4102 }

```

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

symboldoc

```

4103 \NewDocumentEnvironment{symboldoc}{ m }{
4104   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4105   \seq_clear:N \l_tmpb_seq
4106   \seq_map_inline:Nn \l_tmpa_seq {
4107     \str_if_eq:nnF{ ##1 }{}{
4108       \stex_get_symbol:n { ##1 }
4109       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4110         \l_stex_get_symbol_uri_str
4111       }
4112     }
4113   }
4114   \par
4115   \exp_args:Nnnx
4116   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4117 }{
4118   \end{stex_annotate_env}
4119 }

4120 \</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

```
4121 <*package>
4122 <@@=stex_sproof>
4123
4124 %%%%%%%%%% sproof.dtx %%%%%%%%%%
4125
```

34.2 Proofs

We first define some keys for the proof environment.

```
4126 \keys_define:nn { stex / spf } {
4127   id          .str_set:N = \l__stex_sproof_spf_id_str,
4128   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4129   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4130   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4131   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4132   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4133   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4134   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4135   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4136   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4137 }
4138 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4139   \str_clear:N \l__stex_sproof_spf_id_str
4140   \tl_clear:N \l__stex_sproof_spf_display_tl
4141   \tl_clear:N \l__stex_sproof_spf_for_tl
4142   \tl_clear:N \l__stex_sproof_spf_from_tl
4143   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4144   \tl_clear:N \l__stex_sproof_spf_type_tl
4145   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4146 \tl_clear:N \l__stex_sproof_spf_continues_tl
4147 \tl_clear:N \l__stex_sproof_spf_functions_tl
4148 \tl_clear:N \l__stex_sproof_spf_method_tl
4149 \keys_set:nn { stex / spf }{ #1 }
4150 }

```

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

```

4151 \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.

```

4152 \newcount\count_ten
4153 \newenvironment{pst@with@label}[1]{
4154   \edef\pst@label{#1}
4155   \advance\count_ten by 1\relax
4156   \count_ten=1
4157 }{
4158   \advance\count_ten by -1\relax
4159 }

```

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

```

4160 \def\the@pst@label{
4161   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4162 }

```

(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.

```

4163 \keys_define:nn { stex / pstlabel }{
4164   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4165   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4166   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4167 }
4168 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4169 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4170 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4171 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4172 }
4173 \__stex_sproof_pstlabel_args:n {}
4174 \newcommand\setpstlabelstyle[1]{
4175   \__stex_sproof_pstlabel_args:n {#1}
4176 }
4177 \newcommand\setpstlabelstyledefault{%
4178   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4179 }

```

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

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

```

4180 \ExplSyntaxOff
4181 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4182 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4183 \def\pst@make@label@short#1#2{#2}
4184 \def\pst@make@label@empty#1#2{}
4185 \ExplSyntaxOn
4186 \def\pstlabelstyle#1{%
4187   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4188 }%
4189 \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.

```

4190 \def\next@pst@label{%
4191   \global\advance\count\count10 by 1%
4192 }%

```

(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

```

4193 \def\sproof@box{
4194   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4195 }
4196 \def\spf@proofend{\sproof@box}
4197 \def\sproofend{
4198   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4199     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4200   }
4201 }
4202 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4203 \def\spf@proofsketch@kw{Proof Sketch}
4204 \def\spf@proof@kw{Proof}
4205 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4206 \cs_if_exist:NT \bbl@loaded {
4207   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4208   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4209     \input{proof-ngerman.ldf}
4210   }
4211   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4212     \input{proof-finnish.ldf}
4213   }
4214   \clist_if_in:NnT \l_tmpa_clist {french}{
4215     \input{proof-french.ldf}
4216   }
4217   \clist_if_in:NnT \l_tmpa_clist {russian}{
4218     \input{proof-russian.ldf}
4219   }
4220 }
4221

```

spfsketch

```

4222 \newcommand\spfsketch[2][]{
4223   \__stex_sproof_spf_args:n{#1}
4224   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4225     \titleemph{
4226       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4227         \spf@proofsketch@kw
4228       }{
4229         \l__stex_sproof_spf_type_tl
4230       }
4231     }:
4232   }
4233   {-#2}
4234   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4235   \sproofend
4236 }

```

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

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

```

4237 \newenvironment{spfeq}[2][]{
4238   \__stex_sproof_spf_args:n{#1}
4239   %\sref@target
4240   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4241     \titleemph{
4242       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4243         \spf@proof@kw
4244       }{
4245         \l__stex_sproof_spf_type_tl
4246       }
4247     }:

```

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

¹⁵EDNOTE: document above

```

4248 }
4249 {~#2}
4250 \begin{displaymath}\begin{array}{rcll}
4251 }{
4252 \end{array}\end{displaymath}
4253 }

```

(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.

```

4254 \newenvironment{spf@proof}[2][]{
4255   \__stex_sproof_spf_args:n{#1}
4256   %\sref@target
4257   \count_ten=10
4258   \par\noindent
4259   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4260     \titleemph{
4261       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4262         \spf@proof@kw
4263       }{
4264         \l__stex_sproof_spf_type_tl
4265       }
4266     }:
4267   }
4268   {~#2}
4269   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4270   \def\pst@label{}
4271   \newcount\pst@count% initialize the labeling mechanism
4272   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4273   }{
4274     \end{pst@with@label}\end{description}
4275   }
4276 \newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4277 \newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4278 \newcommand\spfidea[2][]{
4279   \__stex_sproof_spf_args:n{#1}
4280   \titleemph{
4281     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4282       \l__stex_sproof_spf_type_tl
4283     }:
4284   }~#2
4285   \sproofend
4286 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep 16

```

4287 \newenvironment{spfstep}[1][]{
4288   \_stex_sproof_spf_args:n{#1}
4289   \@in@omtexttrue
4290   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4291     \item[\the@pst@label]
4292   }
4293   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4294     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4295   }
4296   %\sref@label@id{\pst@label}
4297   \ignorespacesandpars
4298 }{
4299   \next@pst@label\ignorespacesandpars
4300 }

```

sproofcomment

```

4301 \newenvironment{sproofcomment}[1][]{
4302   \_stex_sproof_spf_args:n{#1}
4303   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4304     \item[\the@pst@label]
4305   }
4306 }{
4307   \next@pst@label
4308 }

```

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.

```

4309 \newenvironment{subproof}[2][]{
4310   \_stex_sproof_spf_args:n{#1}
4311   \def\@test{#2}
4312   \ifx\@test\empty\else
4313     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4314       \item[\the@pst@label]
4315     }{#2}
4316   \fi
4317   \begin{pst@with@label}{\pst@label,\number\count_ten}
4318 }{
4319   \end{pst@with@label}\next@pst@label
4320 }

```

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

```

4321 \newenvironment{spfcases}[2][]{
4322   \def\@test{#1}
4323   \ifx\@test\empty
4324     \begin{subproof}[method=by-cases]{#2}
4325   \else
4326     \begin{subproof}[#1,method=by-cases]{#2}
4327   \fi
4328 }{

```

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

```

4329 \end{subproof}
4330 }

```

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

```

4331 \newenvironment{spfcase}[2] [] {
4332   \__stex_sproof_spf_args:n{#1}
4333   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4334     \item[\the@pst@label]
4335   }
4336   \def\@test{#2}
4337   \ifx\@test\@empty
4338   \else
4339     {\titleemph{#2}:~}
4340   \fi
4341   \begin{pst@with@label}{\pst@label,\number\count_ten}
4342 }{
4343   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4344     \sproofend
4345   }
4346   \end{pst@with@label}
4347   \next@pst@label
4348 }

```

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

```

4349 \newcommand\spfcasesketch[3] [] {
4350   \__stex_sproof_spf_args:n{#1}
4351   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4352     \item[\the@pst@label]
4353   }
4354   \def\@test{#2}
4355   \ifx\@test\@empty
4356   \else
4357     {\titleemph{#2}:~}
4358   \fi#3
4359   \next@pst@label
4360 }%

```

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.

```

4361 \keys_define:nn { stex / just }{
4362   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4363   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
4364   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
4365   args        .tl_set:N   = \l__stex_sproof_just_args_tl
4366 }

```

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

```
4367 \newenvironment{justification}[1] [] {}{}
```

\premise

```
4368 \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.

```
4369 \newcommand\justarg[2] [] {#2}
```

```
4370 \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

```
4371 <*package>
4372
4373 %%%%%%%%%% others.dtx %%%%%%%%%%
4374
4375 <@@=stex_others>
    Warnings and error messages
4376 % None

\MSC Math subject classifier

4377 \NewDocumentCommand \MSC {m} {
4378 % TODO
4379 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4380 \@ifpackageloaded{tikzinput}{
4381 \RequirePackage{stex-tikzinput}
4382 }{}
4383 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4384 <*package>
4385 <@@=stex_modules>
4386
4387 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4388
4389 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4390 \begingroup
4391 \stex_module_setup:nn{
4392   ns=\c_stex_metatheory_ns_str,
4393   meta=NONE
4394 }{Metatheory}
4395 \stex_reactivate_macro:N \symdecl
4396 \stex_reactivate_macro:N \notation
4397 \stex_reactivate_macro:N \symdef
4398 \ExplSyntaxOff
4399 \csname stex_suppress_html:n\endcsname{
4400   % is-a (a:A, a \in A, a is an A, etc.)
4401   \symdecl[args=ai]{isa}
4402   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4403   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4404   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4405
4406   % bind (\forall, \Pi, \lambda etc.)
4407   \symdecl[args=Bi]{bind}
4408   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4409   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4410   \notation[depfun]{bind}{\comp( #1 \comp)\;\to\;}{#1 \comp, #2}
4411
4412   % dummy variable
4413   \symdecl{dummyvar}
4414   \notation[underscore]{dummyvar}{\comp\_}
4415   \notation[dot]{dummyvar}{\comp\cdot}
4416   \notation[dash]{dummyvar}{\comp{\rm --}}
4417
4418   %fromto (function space, Hom-set, implication etc.)
```

```

4419 \symdecl[args=ai]{fromto}
4420 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4421 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4422
4423 % mapto (lambda etc.)
4424 %\symdecl[args=Bi]{mapto}
4425 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4426 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4427 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4428
4429 % function/operator application
4430 \symdecl[args=ia]{apply}
4431 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4432 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4433
4434 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4435 \symdecl{collection}
4436 \notation[U]{collection}{\comp{\mathcal{U}}}
4437 \notation[set]{collection}{\comp{\textsf{Set}}}
4438
4439 % sequences
4440 \symdecl[args=1]{seqtype}
4441 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4442
4443 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4444 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4445
4446 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4447 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4448 % ^ superceded by \aseqfromto and \livar/\uivar
4449
4450 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4451 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4452 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4453
4454 % letin (‘‘let’’, local definitions, variable substitution)
4455 \symdecl[args=bii]{letin}
4456 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4457 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4458 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4459
4460 % structures
4461 \symdecl*[args=1]{module-type}
4462 \notation{module-type}{\mathtt{MOD} #1}
4463 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4464 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4465
4466 }
4467 \ExplSyntaxOn
4468 \stex_add_to_current_module:n{
4469   \let\nappa\apply
4470   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4471   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4472   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4473 \def\uivar{\csname sequence-index\endcsname[ui]}
4474 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4475 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4476 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4477 }
4478 \__stex_modules_end_module:
4479 \endgroup
4480 \</package>

```

Chapter 37

Tikzinput Implementation

```
4481 <*package>
4482
4483 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4484
4485 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4486 \RequirePackage{l3keys2e}
4487
4488 \keys_define:nn { tikzinput } {
4489   image .bool_set:N = \c_tikzinput_image_bool,
4490   image .default:n = false ,
4491   unknown .code:n = {}
4492 }
4493
4494 \ProcessKeysOptions { tikzinput }
4495
4496 \bool_if:NTF \c_tikzinput_image_bool {
4497   \RequirePackage{graphicx}
4498
4499   \providecommand\usetikzlibrary[]{}
4500   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4501 }{
4502   \RequirePackage{tikz}
4503   \RequirePackage{standalone}
4504
4505   \newcommand \tikzinput [2] [] {
4506     \setkeys{Gin}{#1}
4507     \ifx \Gin@ewidth \Gin@exclamation
4508       \ifx \Gin@eheight \Gin@exclamation
4509         \input { #2 }
4510       \else
4511         \resizebox{!}{ \Gin@eheight }{
4512           \input { #2 }
4513         }
4514       \fi
4515     \else
4516       \ifx \Gin@eheight \Gin@exclamation
4517         \resizebox{ \Gin@ewidth }{!}{
4518           \input { #2 }
```



```

4519     }
4520     \else
4521     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4522     \input { #2 }
4523     }
4524     \fi
4525     \fi
4526   }
4527 }
4528
4529 \newcommand \ctikzinput [2] [] {
4530   \begin{center}
4531     \tikzinput [1] {#2}
4532   \end{center}
4533 }
4534
4535 \@ifpackageloaded{stex}{
4536   \RequirePackage{stex-tikzinput}
4537 }{}
4538
4539 </package>
4540 <*stex>
4541 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4542 \RequirePackage{stex}
4543 \RequirePackage{tikzinput}
4544
4545 \newcommand\mhtikzinput [2] [] {%
4546   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4547   \stex_in_repository:nn\Gin@mhrepos{
4548     \tikzinput [1]{\mhpath{##1}{#2}}
4549   }
4550 }
4551 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4552 </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.

```
4553 \*cls)
4554 \@@=document_structure)
4555 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4556 \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`

```
4557 \keys_define:nn{ document-structure / pkg }{
4558   class      .str_set_x:N = \c_document_structure_class_str,
4559   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4560   report     .code:n      = {
4561     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4562     \str_set:Nn \c_document_structure_class_str {report}
4563   },
4564   book       .code:n      = {
4565     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4566     \str_set:Nn \c_document_structure_class_str {book}
4567   },
4568   bookpart   .code:n      = {
4569     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4570     \str_set:Nn \c_document_structure_class_str {book}
4571     \str_set:Nn \c_document_structure_topsect_str {chapter}
4572   },
```

```

4573 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4574 unknown     .code:n      = {
4575   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4576 }
4577 }
4578 \ProcessKeysOptions{ document-structure / pkg }
4579 \str_if_empty:NT \c_document_structure_class_str {
4580   \str_set:Nn \c_document_structure_class_str {article}
4581 }
4582 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4583   {\c_document_structure_class_str}
4584

```

38.3 Beefing up the document environment

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

```

4585 \RequirePackage{omdoc}
4586 \bool_if:NF \c_document_structure_minimal_bool {
4587   \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

```

4588 \keys_define:nn { document-structure / document }{
4589   id .str_set_x:N = \c_document_structure_document_id_str
4590 }
4591 \let\__document_structure_orig_document=\document
4592 \renewcommand{\document}[1][]{
4593   \keys_set:nn{ document-structure / document }{ #1 }
4594   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4595   \__document_structure_orig_document
4596 }

```

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

```

4597 }
4598 \</cls>

```

38.4 Implementation: OMDoc Package

```

4599 \*package>
4600 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4601 \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

```

4602
4603 \keys_define:nn{ document-structure / pkg }{
4604   class      .str_set_x:N = \c_document_structure_class_str,
4605   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4606   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4607 }
4608 \ProcessKeysOptions{ document-structure / pkg }
4609 \str_if_empty:NT \c_document_structure_class_str {
4610   \str_set:Nn \c_document_structure_class_str {article}
4611 }
4612 \str_if_empty:NT \c_document_structure_topsect_str {
4613   \str_set:Nn \c_document_structure_topsect_str {section}
4614 }

```

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

```

4615 \RequirePackage{xspace}
4616 \RequirePackage{comment}
4617 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4618 \@ifpackageloaded{babel}{
4619   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4620   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4621     \input{omdoc-ngerman.ldf}
4622   }
4623 }{}
4624 %\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.

```

4625 \int_new:N \l_document_structure_section_level_int
4626 \str_case:VnF \c_document_structure_topsect_str {
4627   {part}{
4628     \int_set:Nn \l_document_structure_section_level_int {0}
4629   }
4630   {chapter}{
4631     \int_set:Nn \l_document_structure_section_level_int {1}
4632   }
4633 }{
4634   \str_case:VnF \c_document_structure_class_str {
4635     {book}{
4636       \int_set:Nn \l_document_structure_section_level_int {0}
4637     }
4638     {report}{
4639       \int_set:Nn \l_document_structure_section_level_int {0}
4640     }
4641   }{
4642     \int_set:Nn \l_document_structure_section_level_int {2}
4643   }
4644 }

```

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

```
4645 \def\current@section@level{document}%
4646 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4647 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4648 \cs_new_protected:Npn \skipomgroup {
4649   \ifcase\l_document_structure_section_level_int
4650   \or\stepcounter{part}
4651   \or\stepcounter{chapter}
4652   \or\stepcounter{section}
4653   \or\stepcounter{subsection}
4654   \or\stepcounter{subsubsection}
4655   \or\stepcounter{paragraph}
4656   \or\stepcounter{subparagraph}
4657   \fi
4658 }
```

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

`blindomgroup`

```
4659 \newcommand\at@begin@blindomgroup[1]{%
4660 \newenvironment{blindomgroup}
4661 {
4662   \int_incr:N\l_document_structure_section_level_int
4663   \at@begin@blindomgroup\l_document_structure_section_level_int
4664 }{}}
```

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

```
4665 \newcommand\omgroup@nonum[2]{
4666   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4667   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4668 }
```

(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.

```
4669 \newcommand\omgroup@num[2]{
```

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

```

4670 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4671   \@nameuse{#1}{#2}
4672 }{
4673   \cs_if_exist:NTF\rdfmata@sectioning{
4674     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4675   }{
4676     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4677   }
4678 }
4679 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4680 }

```

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

omgroup

```

4681 \keys_define:nn { document-structure / omgroupp }{
4682   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4683   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4684   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
4685   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4686   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
4687   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
4688   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
4689   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
4690   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
4691   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
4692 }
4693 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4694   \str_clear:N \l__document_structure_omgroup_id_str
4695   \str_clear:N \l__document_structure_omgroup_date_str
4696   \clist_clear:N \l__document_structure_omgroup_creators_clist
4697   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4698   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4699   \tl_clear:N \l__document_structure_omgroup_type_tl
4700   \tl_clear:N \l__document_structure_omgroup_short_tl
4701   \tl_clear:N \l__document_structure_omgroup_display_tl
4702   \tl_clear:N \l__document_structure_omgroup_intro_tl
4703   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4704   \keys_set:nn { document-structure / omgroupp } { #1 }
4705 }

```

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 omgroupp, i.e. after the section heading.

```

4706 \newif\if@mainmatter\@mainmattertrue
4707 \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.

```

4708 \keys_define:nn { document-structure / sectioning }{
4709   name .str_set_x:N = \l__document_structure_sect_name_str ,
4710   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4711   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4712   num .bool_set:N = \l__document_structure_sect_num_bool ,
4713 }

```

```

4714 \cs_new_protected:Nn \__document_structure_sect_args:n {
4715   \str_clear:N \l__document_structure_sect_name_str
4716   \str_clear:N \l__document_structure_sect_ref_str
4717   \bool_set_false:N \l__document_structure_sect_clear_bool
4718   \bool_set_false:N \l__document_structure_sect_num_bool
4719   \keys_set:nn { document-structure / sectioning } { #1 }
4720 }
4721 \newcommand\omdoc@sectioning[3][]{
4722   \__document_structure_sect_args:n {#1}
4723   \let\omdoc@sect@name\l__document_structure_sect_name_str
4724   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4725   \if@mainmatter% numbering not overridden by frontmatter, etc.
4726     \bool_if:NTF \l__document_structure_sect_num_bool {
4727       \omgroup@num{#2}{#3}
4728     }{
4729       \omgroup@nonum{#2}{#3}
4730     }
4731     \def\current@section@level{\omdoc@sect@name}
4732   \else
4733     \omgroup@nonum{#2}{#3}
4734   \fi
4735 }% 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.

```

4736 \newcommand\omgroup@redefine@addtocontents[1]{%
4737   %\edef\__document_structureimport{#1}%
4738   %\@for\@I:=\__document_structureimport\do{%
4739     %\edef\@path{\csname module@\@I @path\endcsname}%
4740     %\@ifundefined{tf@toc}\relax%
4741     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4742   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4743   %\def\addcontentsline##1##2##3{%
4744     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4745   %\else% hyperref.sty not loaded
4746   %\def\addcontentsline##1##2##3{%
4747     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4748   %\fi
4749 }% 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.

```

4750 \int_new:N \l_document_structure_omgroup_level_int
4751 \newenvironment{omgroup}[2][]{% keys, title
4752 {
4753   \__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.

```

4754 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4755   \omgroup@redefine@addtocontents{
4756     %\@ifundefined{module@id}\used@modules%
4757     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4758     }
4759 }

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

4760 \int_incr:N \l_document_structure_omgroup_level_int
4761 \int_incr:N \l_document_structure_section_level_int
4762 \ifcase\l_document_structure_section_level_int
4763   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4764   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4765   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4766   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4767   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4768   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4769   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4770 \fi
4771 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4772 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4773 }% for customization
4774 {}

```

and finally, we localize the sections

```

4775 \newcommand\omdoc@part@kw{Part}
4776 \newcommand\omdoc@chapter@kw{Chapter}
4777 \newcommand\omdoc@section@kw{Section}
4778 \newcommand\omdoc@subsection@kw{Subsection}
4779 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4780 \newcommand\omdoc@paragraph@kw{paragraph}
4781 \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`

```

4782 \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).

```

4783 \cs_if_exist:NTF\frontmatter{
4784   \let\__document_structure_orig_frontmatter\frontmatter
4785   \let\frontmatter\relax
4786 }{
4787   \tl_set:Nn\__document_structure_orig_frontmatter{
4788     \clearpage
4789     \@mainmatterfalse
4790     \pagenumbering{roman}
4791   }
4792 }
4793 \cs_if_exist:NTF\backmatter{

```



```

4794 \let\__document_structure_orig_backmatter\backmatter
4795 \let\backmatter\relax
4796 }{
4797 \tl_set:Nn\__document_structure_orig_backmatter{
4798 \clearpage
4799 \@mainmatterfalse
4800 \pagenumbering{roman}
4801 }
4802 }

```

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.

```

4803 \newenvironment{frontmatter}{
4804 \__document_structure_orig_frontmatter
4805 }{
4806 \cs_if_exist:NTF\mainmatter{
4807 \mainmatter
4808 }{
4809 \clearpage
4810 \@mainmattertrue
4811 \pagenumbering{arabic}
4812 }
4813 }

```

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

```

4814 \newenvironment{backmatter}{
4815 \__document_structure_orig_backmatter
4816 }{
4817 \cs_if_exist:NTF\mainmatter{
4818 \mainmatter
4819 }{
4820 \clearpage
4821 \@mainmattertrue
4822 \pagenumbering{arabic}
4823 }
4824 }

```

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

```

4825 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4826 \def \c__document_structure_document_str{document}
4827 \newcommand\afterprematurestop{}
4828 \def\prematurestop@endomgroup{
4829 \unless\ifx\@currenvir\c__document_structure_document_str
4830 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4831 \expandafter\prematurestop@endomgroup
4832 \fi
4833 }
4834 \providecommand\prematurestop{

```

```

4835 \message{Stopping~sTeX~processing~prematurely}
4836 \prematurestop@endomgroup
4837 \afterprematurestop
4838 \end{document}
4839 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

4840 \RequirePackage{etoolbox}
4841 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4842 \newrobustcmd\useSGvar[1]{%
4843 \@ifundefined{sTeX@Gvar@#1}
4844 {\PackageError{omdoc}
4845 {The sTeX Global variable #1 is undefined}
4846 {set it with \protect\setSGvar}}
4847 \@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.

```

4848 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4849 \@ifundefined{sTeX@Gvar@#1}
4850 {\PackageError{omdoc}
4851 {The sTeX Global variable #1 is undefined}
4852 {set it with \protect\setSGvar}}
4853 {\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.

```
4854 \*cls)
4855 \@@=mikoslides)
4856 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4857 \RequirePackage{l3keys2e,expl-keystr-compatible}
4858
4859 \keys_define:nn{mikoslides / cls}{
4860   class .code:n = {
4861     \PassOptionsToClass{\CurrentOption}{omdoc}
4862     \str_if_eq:nnT{#1}{book}{
4863       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4864     }
4865     \str_if_eq:nnT{#1}{report}{
4866       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4867     }
4868   },
4869   notes .bool_set:N = \c__mikoslides_notes_bool ,
4870   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4871   unknown .code:n = {
4872     \PassOptionsToClass{\CurrentOption}{omdoc}
4873     \PassOptionsToClass{\CurrentOption}{beamer}
4874     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4875   }
4876 }
4877 \ProcessKeysOptions{ mikoslides / cls }
4878 \bool_if:NTF \c__mikoslides_notes_bool {
4879   \PassOptionsToPackage{notes=true}{mikoslides}
4880 }{
4881   \PassOptionsToPackage{notes=false}{mikoslides}
4882 }
4883 \</cls)
```

now we do the same for the mikoslides package.

```

4884 \*package>
4885 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4886 \RequirePackage{l3keys2e,expl-keystr-compat}
4887
4888 \keys_define:nn{mikoslides / pkg}{
4889   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4890   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4891   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4892   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4893   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4894   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4895   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4896   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4897   unknown      .code:n      = {
4898     \PassOptionsToClass{\CurrentOption}{stex}
4899     \PassOptionsToClass{\CurrentOption}{tikzinput}
4900   }
4901 }
4902 \ProcessKeysOptions{ mikoslides / pkg }
4903 \newif\ifnotes
4904 \bool_if:NTF \c__mikoslides_notes_bool {
4905   \notesttrue
4906 }{
4907   \notesfalse
4908 }
4909

```

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

```

4910 \str_if_empty:NTF \c__mikoslides_topsect_str {
4911   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4912 }{
4913   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4914 }
4915 \</package>

```

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

```

4916 \*cls>
4917 \bool_if:NTF \c__mikoslides_notes_bool {
4918   \LoadClass{omdoc}
4919 }{
4920   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4921   \newcounter{Item}
4922   \newcounter{paragraph}
4923   \newcounter{subparagraph}
4924   \newcounter{Hfootnote}
4925   \RequirePackage{omdoc}
4926 }

```

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

```

4927 \RequirePackage{mikoslides}
4928 \</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.

```

4929 \*package>
4930 \bool_if:NT \c__mikoslides_notes_bool {
4931   \RequirePackage{a4wide}
4932   \RequirePackage{marginnote}
4933   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4934   \RequirePackage{mdframed}
4935   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4936   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4937 }
4938 \RequirePackage{stex-compatibility}
4939 \RequirePackage{stex-tikzinput}
4940 \RequirePackage{etoolbox}
4941 \RequirePackage{amssymb}
4942 \RequirePackage{amsmath}
4943 \RequirePackage{comment}
4944 \RequirePackage{textcomp}
4945 \RequirePackage{url}
4946 \RequirePackage{graphicx}
4947 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.²⁰

```

4948 \bool_if:NT \c__mikoslides_notes_bool {
4949   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4950 }

```

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

```

4951 \newcounter{slide}
4952 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4953 \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.

```

4954 \bool_if:NTF \c__mikoslides_notes_bool {
4955   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4956 }{
4957   \excludecomment{note}
4958 }

```

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.

```
4959 \bool_if:NT \c__mikoslides_notes_bool {
4960   \newlength{\slideframewidth}
4961   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4962 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4963   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4964     \bool_set_true:N #1
4965   }{
4966     \bool_set_false:N #1
4967   }
4968 }
4969 \keys_define:nn{mikoslides / frame}{
4970   label .str_set_x:N = \l__mikoslides_frame_label_str,
4971   allowframebreaks .code:n = {
4972     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4973   },
4974   allowdisplaybreaks .code:n = {
4975     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4976   },
4977   fragile .code:n = {
4978     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4979   },
4980   shrink .code:n = {
4981     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4982   },
4983   squeeze .code:n = {
4984     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4985   },
4986   t .code:n = {
4987     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4988   },
4989 }
4990 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4991   \str_clear:N \l__mikoslides_frame_label_str
4992   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4993   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4994   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4995   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4996   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4997   \bool_set_true:N \l__mikoslides_frame_t_bool
4998   \keys_set:nn { mikoslides / frame }{ #1 }
4999 }
```

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

```
5000 \renewenvironment{frame}[1][]{
5001   \__mikoslides_frame_args:n{#1}
5002   \sffamily
5003   \stepcounter{slide}
5004   \def\@currentlabel{\theslide}
5005   \str_if_empty:NF \l__mikoslides_frame_label_str {
5006     \label{\l__mikoslides_frame_label_str}
```

```
5007 }
```

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

```
5008 \def\itemize@level{outer}
5009 \def\itemize@outer{outer}
5010 \def\itemize@inner{inner}
5011 \renewcommand\newpage{\addtocounter{framenum}{1}}
5012 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5013 \renewenvironment{itemize}{
5014   \ifx\itemize@level\itemize@outer
5015     \def\itemize@label{\$ \rhd \$}
5016   \fi
5017   \ifx\itemize@level\itemize@inner
5018     \def\itemize@label{\$ \scriptstyle \rhd \$}
5019   \fi
5020   \begin{list}
5021     {\itemize@label}
5022     {\setlength{\labelsep}{.3em}
5023      \setlength{\labelwidth}{.5em}
5024      \setlength{\leftmargin}{1.5em}
5025     }
5026   \edef\itemize@level{\itemize@inner}
5027 }{
5028   \end{list}
5029 }
```

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

```
5030 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5031 }{
5032   \medskip\miko@slidelabel\end{mdframed}
5033 }
```

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

`\frametitle`

```
5034 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5035 }
```

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

EdN:21

`\pause` 21

```
5036 \bool_if:NT \c__mikoslides_notes_bool {
5037   \newcommand\pause{}
5038 }
```

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

`nomtext`

```
5039 \bool_if:NTF \c__mikoslides_notes_bool {
5040   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5041 }{
5042   \excludecomment{nomtext}
5043 }
```

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

nomgroup

```
5044 \bool_if:NTF \c__mikoslides_notes_bool {
5045   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5046 }{
5047   \excludecomment{nomgroup}
5048 }
```

ndefinition

```
5049 \bool_if:NTF \c__mikoslides_notes_bool {
5050   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5051 }{
5052   \excludecomment{ndefinition}
5053 }
```

nassertion

```
5054 \bool_if:NTF \c__mikoslides_notes_bool {
5055   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5056 }{
5057   \excludecomment{nassertion}
5058 }
```

nsproof

```
5059 \bool_if:NTF \c__mikoslides_notes_bool {
5060   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5061 }{
5062   \excludecomment{nproof}
5063 }
```

nexample

```
5064 \bool_if:NTF \c__mikoslides_notes_bool {
5065   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5066 }{
5067   \excludecomment{nexample}
5068 }
```

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

```
5069 \def\inputref@preskip{\smallskip}
5070 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5071 \let\orig@inputref\inputref
5072 \def\inputref{\@ifstar\ninputref\orig@inputref}
5073 \newcommand\ninputref[2] [] {
5074   \bool_if:NT \c__mikoslides_notes_bool {
5075     \orig@inputref[#1]{#2}
5076   }
5077 }
```

(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>}`.

```
5078 \newlength{\slidelogoheight}
5079
5080 \bool_if:NTF \c__mikoslides_notes_bool {
5081   \setlength{\slidelogoheight}{.4cm}
5082 }{
5083   \setlength{\slidelogoheight}{1cm}
5084 }
5085 \newsavebox{\slidelogo}
5086 \sbox{\slidelogo}{\TeX}
5087 \newrobustcmd{\setslidelogo}[1]{
5088   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5089 }
```

(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.

```
5090 \def\source{Michael Kohlhase}% customize locally
5091 \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.

```
5092 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5093 \newsavebox{\cclogo}
5094 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5095 \newif\ifcchref\cchreffalse
5096 \AtBeginDocument{
5097   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5098 }
5099 \def\licensing{
5100   \ifcchref
5101     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5102   \else
5103     {\usebox{\cclogo}}
5104   \fi
5105 }
5106 \newrobustcmd{\setlicensing}[2][]{
5107   \def\@url{#1}
5108   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5109   \ifx\@url\@empty
5110     \def\licensing{{\usebox{\cclogo}}}
5111   \else
5112     \def\licensing{
```

```

5113     \ifcchref
5114     \href{#1}{\usebox{\cclogo}}
5115     \else
5116     {\usebox{\cclogo}}
5117     \fi
5118   }
5119   \fi
5120 }

```

(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

```

5121 \newrobustcmd\miko@slidelabel{
5122   \vbox to \slidelogoheight{
5123     \vss\hbox to \slidewidth
5124     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5125   }
5126 }

```

(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.

```

5127 \def\Gin@mhrepos{}
5128 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5129 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5130 \newrobustcmd\frameimage[2][]{
5131   \stepcounter{slide}
5132   \bool_if:NT \c__mikoslides_frameimages_bool {
5133     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5134     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5135     \begin{center}
5136       \bool_if:NTF \c__mikoslides_fiboxed_bool {
5137         \fbox{
5138           \ifx\Gin@ewidth\@empty
5139             \ifx\Gin@mhrepos\@empty
5140               \mhgraphics[width=\slidewidth,#1]{#2}
5141             \else
5142               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5143             \fi
5144           \else% \Gin@ewidth empty
5145             \ifx\Gin@mhrepos\@empty
5146               \mhgraphics[#1]{#2}
5147             \else
5148               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5149             \fi
5150           \fi% \Gin@ewidth empty
5151         }
5152       }{
5153         \ifx\Gin@ewidth\@empty

```

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

```

5154         \ifx\Gin@mhrepos\empty
5155             \mhgraphics[width=\slidewidth,#1]{#2}
5156         \else
5157             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5158         \fi
5159         \ifx\Gin@mhrepos\empty
5160             \mhgraphics[#1]{#2}
5161         \else
5162             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5163         \fi
5164     \fi% Gin@ewidth empty
5165 }
5166 \end{center}
5167 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5168 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5169 }
5170 } % 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.

```

5171 \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

```

5172 \AddToHook{begindocument}{
5173     \definecolor{green}{rgb}{0,.5,0}
5174     \definecolor{purple}{cmyk}{.3,1,0,.17}
5175 }

```

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`.

```

5176 % \def\STpresent#1{\textcolor{blue}{#1}}
5177 \def\defemph#1{\textcolor{magenta}{#1}}
5178 \def\symrefemph#1{\textcolor{cyan}{#1}}
5179 \def\compemph#1{\textcolor{blue}{#1}}
5180 \def\titleemph#1{\textcolor{blue}{#1}}
5181 \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.

```

5182 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5183 \def\smalltextwarning{
5184     \pgfuseimage{miko@small@dbend}
5185     \xspace
5186 }
5187 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5188 \newrobustcmd\textwarning{
5189   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5190   \xspace
5191 }
5192 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5193 \newrobustcmd\bigtextwarning{
5194   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5195   \xspace
5196 }

```

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

```

5197 \newrobustcmd\putgraphicsat[3]{
5198   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5199 }
5200 \newrobustcmd\putat[2]{
5201   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5202 }

```

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`.

```

5203 \bool_if:NT \c__mikoslides_sectocframes_bool {
5204   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5205     \newcounter{chapter}\counterwithin*{section}{chapter}
5206   }{
5207     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5208       \newcounter{chapter}\counterwithin*{section}{chapter}
5209     }
5210   }
5211 }

```

`\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
5212 \def\part@prefix{}
5213 \@ifpackageloaded{omdoc}{}{
5214   \str_case:VnF \__mikoslidestopsect {
5215     {part}{
5216       \int_set:Nn \l_document_structure_section_level_int {0}
5217       \def\thesection{\arabic{chapter}.\arabic{section}}
5218       \def\part@prefix{\arabic{chapter}.}
5219     }
5220     {chapter}{
5221       \int_set:Nn \l_document_structure_section_level_int {1}
5222       \def\thesection{\arabic{chapter}.\arabic{section}}
5223       \def\part@prefix{\arabic{chapter}.}
5224     }
5225   }{
5226     \int_set:Nn \l_document_structure_section_level_int {2}
5227     \def\part@prefix{}

```

```

5228 }
5229 }
5230
5231 \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

```

5232 \renewenvironment{omgroup}[2][]{
5233   \__document_structure_omgroup_args:n { #1 }
5234   \int_incr:N \l_document_structure_omgroup_level_int
5235   \int_incr:N \l_document_structure_section_level_int
5236   \bool_if:NT \c__mikoslides_sectocframes_bool {
5237     \stepcounter{slide}
5238     \begin{frame}[noframenumbering]
5239     \vfill\Large\centering
5240     \red{
5241       \ifcase\l_document_structure_section_level_int\or
5242         \stepcounter{part}
5243         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5244         \def\currentsectionlevel{\omdoc@part@kw}
5245       \or
5246         \stepcounter{chapter}
5247         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5248         \def\currentsectionlevel{\omdoc@chapter@kw}
5249       \or
5250         \stepcounter{section}
5251         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5252         \def\currentsectionlevel{\omdoc@section@kw}
5253       \or
5254         \stepcounter{subsection}
5255         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5256         \def\currentsectionlevel{\omdoc@subsection@kw}
5257       \or
5258         \stepcounter{subsubsection}
5259         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5260         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5261       \or
5262         \stepcounter{paragraph}
5263         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5264         \def\currentsectionlevel{\omdoc@paragraph@kw}
5265       \else
5266         \def\__mikoslideslabel{}
5267         \def\currentsectionlevel{\omdoc@paragraph@kw}
5268       \fi% end ifcase
5269       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5270       \quad #2%
5271     }%
5272     \vfill%
5273     \end{frame}%
5274   }
5275   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5276 }{}
5277 }

```

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

```

5278 \def\inserttheorembodyfont{\normalfont}
5279 %\bool_if:NF \c__mikoslides_notes_bool {
5280 % \defbeamertemplate{theorem begin}{miko}
5281 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5282 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5283 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5284 % \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5285 % \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.

```

5286 % \expandafter\def\csname Parent2\endcsname{}
5287 %}
5288
5289 \AddToHook{begindocument}{% this does not work for some reasons
5290 \setbeamertemplate{theorems}[ams style]
5291 }
5292 \bool_if:NT \c__mikoslides_notes_bool {
5293 \renewenvironment{columns}[1][{}%
5294 \par\noindent%
5295 \begin{minipage}%
5296 \slidewidth\centering\leavevmode%
5297 }{}%
5298 \end{minipage}\par\noindent%
5299 }%
5300 \newsavebox\columnbox%
5301 \renewenvironment<>{column}[2][{}%
5302 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5303 }{}%
5304 \end{minipage}\end{lrbox}\usebox\columnbox%
5305 }%
5306 }
5307 \bool_if:NTF \c__mikoslides_noproblems_bool {
5308 \newenvironment{problems}{}{}
5309 }{
5310 \excludecomment{problems}
5311 }

```

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.

```

5312 \gdef\printexcursions{}
5313 \newcommand\excursionref[2]{% label, text
5314 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5315 \begin{sparagraph}[title=Excursion]
5316 #2 \sref[fallback=the appendix]{#1}.
5317 \end{sparagraph}
5318 }
5319 }
5320 \newcommand\activate@excursion[2][]{
5321 \gappto\printexcursions{\inputref[#1]{#2}}
5322 }
5323 \newcommand\excursion[4][]{% repos, label, path, text
5324 \bool_if:NT \c__mikoslides_notes_bool {
5325 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5326 }
5327 }

```

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

\excursiongroup

```

5328 \keys_define:nn{mikoslides / excursiongroup }{
5329 id .str_set_x:N = \l__mikoslides_excursion_id_str,
5330 intro .tl_set:N = \l__mikoslides_excursion_intro_tl,
5331 mhrepos .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5332 }
5333 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5334 \tl_clear:N \l__mikoslides_excursion_intro_tl
5335 \str_clear:N \l__mikoslides_excursion_id_str
5336 \str_clear:N \l__mikoslides_excursion_mhrepos_str
5337 \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5338 }
5339 \newcommand\excursiongroup[1][]{
5340 \__mikoslides_excursion_args:n{ #1 }
5341 \ifdefempty\printexcursions{}% only if there are excursions
5342 {\begin{note}
5343 \begin{omgroup}[#1]{Excursions}%
5344 \ifdefempty\l__mikoslides_excursion_intro_tl{{
5345 \inputref[\l__mikoslides_excursion_mhrepos_str]{
5346 \l__mikoslides_excursion_intro_tl
5347 }
5348 }
5349 \printexcursions%
5350 \end{omgroup}
5351 \end{note}}
5352 }
5353 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5354 \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.

```
5355 <*package>
5356 <@@=problems>
5357 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5358 \RequirePackage{l3keys2e,expl-keystr-compatible}
5359
5360 \keys_define:nn { problem / pkg }{
5361   notes      .default:n    = { true },
5362   notes      .bool_set:N   = \c__problems_notes_bool,
5363   gnotes     .default:n    = { true },
5364   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5365   hints      .default:n    = { true },
5366   hints      .bool_set:N   = \c__problems_hints_bool,
5367   solutions  .default:n    = { true },
5368   solutions  .bool_set:N   = \c__problems_solutions_bool,
5369   pts        .default:n    = { true },
5370   pts        .bool_set:N   = \c__problems_pts_bool,
5371   min        .default:n    = { true },
5372   min        .bool_set:N   = \c__problems_min_bool,
5373   boxed      .default:n    = { true },
5374   boxed      .bool_set:N   = \c__problems_boxed_bool,
5375   unknown    .code:n       = {}
5376 }
5377 \def\solutionstrue{
5378   \bool_set_true:N \c__problems_solutions_bool
5379 }
5380 \def\solutionsfalse{
5381   \bool_set_false:N \c__problems_solutions_bool
5382 }
5383
5384 \ProcessKeysOptions{ problem / pkg }
```

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


```

5385 \RequirePackage{stex-compatibility}
5386 \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.

```

5387 \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.

```

5388 \def\prob@problem@kw{Problem}
5389 \def\prob@solution@kw{Solution}
5390 \def\prob@hint@kw{Hint}
5391 \def\prob@note@kw{Note}
5392 \def\prob@gnote@kw{Grading}
5393 \def\prob@pt@kw{pt}
5394 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5395 \@ifpackageloaded{babel}{
5396   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5397   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5398     \input{problem-ngerman.ldf}
5399   }
5400   \clist_if_in:NnT \l_tmpa_clist {finnish}{
5401     \input{problem-finnish.ldf}
5402   }
5403   \clist_if_in:NnT \l_tmpa_clist {french}{
5404     \input{problem-french.ldf}
5405   }
5406   \clist_if_in:NnT \l_tmpa_clist {russian}{
5407     \input{problem-russian.ldf}
5408   }
5409 }{}

```

40.2 Problems and Solutions

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

```

5410 \keys_define:nn{ problem / problem }{
5411   id      .str_set:x:N = \l__problems_prob_id_str,
5412   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5413   min     .tl_set:N    = \l__problems_prob_min_tl,
5414   title   .tl_set:N    = \l__problems_prob_title_tl,
5415   refnum  .int_set:N   = \l__problems_prob_refnum_int
5416 }
5417 \cs_new_protected:Nn \__problems_prob_args:n {
5418   \str_clear:N \l__problems_prob_id_str
5419   \tl_clear:N \l__problems_prob_pts_tl
5420   \tl_clear:N \l__problems_prob_min_tl
5421   \tl_clear:N \l__problems_prob_title_tl

```

```

5422 \int_zero_new:N \l__problems_prob_refnum_int
5423 \keys_set:nn { problem / problem }{ #1 }
5424 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5425   \let\l__problems_inclprob_refnum_int\undefined
5426 }
5427 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5428 \newcounter{problem}
5429 \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.

```

5430 \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

```

5431 \newcommand\prob@number{
5432   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5433     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5434   }{
5435     \int_if_exist:NTF \l__problems_prob_refnum_int {
5436       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5437     }{
5438       \prob@label\theproblem
5439     }
5440   }
5441 }

```

(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.

```

5442 \newcommand\prob@title[3]{%
5443   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5444     #2 \l__problems_inclprob_title_tl #3
5445   }{
5446     \tl_if_exist:NTF \l__problems_prob_title_tl {
5447       #2 \l__problems_prob_title_tl #3
5448     }{
5449       #1
5450     }
5451   }
5452 }

```

(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.

```
5453 \def\prob@heading{
5454   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5455   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5456 }
```

(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`

```
5457 \newenvironment{problem}[1][1]{
5458   \__problems_prob_args:n{#1}%\sref@target%
5459   \@in@omtexttrue% we are in a statement (for inline definitions)
5460   \stepcounter{problem}\record@problem
5461   \def\current@section@level{\prob@problem@kw}
5462   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5463 }%
5464 {\smallskip}
5465 \bool_if:NT \c__problems_boxed_bool {
5466   \surroundwithmdframed{problem}
5467 }
```

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

```
5468 \def\record@problem{
5469   \protected@write\@auxout{}
5470   {
5471     \string\@problem{\prob@number}
5472     {
5473       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5474         \l__problems_inclprob_pts_tl
5475       }{
5476         \l__problems_prob_pts_tl
5477       }
5478     }%
5479     {
5480       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5481         \l__problems_inclprob_min_tl
5482       }{
5483         \l__problems_prob_min_tl
5484       }
5485     }
5486   }
5487 }
```

(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).

```
5488 \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.

```

5489 \keys_define:nn { problem / solution }{
5490   id          .str_set_x:N = \l__problems_solution_id_str ,
5491   for         .tl_set:N    = \l__problems_solution_for_tl ,
5492   height      .dim_set:N   = \l__problems_solution_height_dim ,
5493   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5494   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5495   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5496 }
5497 \cs_new_protected:Nn \__problems_solution_args:n {
5498   \str_clear:N \l__problems_solution_id_str
5499   \tl_clear:N \l__problems_solution_for_tl
5500   \tl_clear:N \l__problems_solution_srccite_tl
5501   \clist_clear:N \l__problems_solution_creators_clist
5502   \clist_clear:N \l__problems_solution_contributors_clist
5503   \dim_zero:N \l__problems_solution_height_dim
5504   \keys_set:nn { problem / solution }{ #1 }
5505 }

```

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

```

5506 \newcommand\@startsolution[1][ ]{
5507   \__problems_solution_args:n { #1 }
5508   \@in@omtexttrue% we are in a statement.
5509   \bool_if:NF \c__problems_boxed_bool { \hrule }
5510   \smallskip\noindent
5511   {\textbf\prob@solution@kw : \enspace}
5512   \begin{small}
5513   \def\current@section@level{\prob@solution@kw}
5514   \ignorespacesandpars
5515 }

```

\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.

```

5516 \newcommand\startsolutions{
5517   \specialcomment{solution}{\@startsolution}{
5518     \bool_if:NF \c__problems_boxed_bool {
5519       \hrule\medskip
5520     }
5521     \end{small}%
5522   }
5523   \bool_if:NT \c__problems_boxed_bool {
5524     \surroundwithmdframed{solution}
5525   }
5526 }

```

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

\stopsolutions

```

5527 \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.

```

5528 \bool_if:NTF \c_problems_solutions_bool {
5529   \startsolutions
5530 }{
5531   \stopsolutions
5532 }

```

exnote

```

5533 \bool_if:NTF \c_problems_notes_bool {
5534   \newenvironment{exnote}[1][]{
5535     \par\smallskip\hrule\smallskip
5536     \noindent\textbf{\prob@note@kw : }\small
5537   }{
5538     \smallskip\hrule
5539   }
5540 }{
5541   \excludecomment{exnote}
5542 }

```

hint

```

5543 \bool_if:NTF \c_problems_notes_bool {
5544   \newenvironment{hint}[1][]{
5545     \par\smallskip\hrule\smallskip
5546     \noindent\textbf{\prob@hint@kw :~ }\small
5547   }{
5548     \smallskip\hrule
5549   }
5550   \newenvironment{exhint}[1][]{
5551     \par\smallskip\hrule\smallskip
5552     \noindent\textbf{\prob@hint@kw :~ }\small
5553   }{
5554     \smallskip\hrule
5555   }
5556 }{
5557   \excludecomment{hint}
5558   \excludecomment{exhint}
5559 }

```

gnote

```

5560 \bool_if:NTF \c_problems_notes_bool {
5561   \newenvironment{gnote}[1][]{
5562     \par\smallskip\hrule\smallskip
5563     \noindent\textbf{\prob@gnote@kw : }\small
5564   }{
5565     \smallskip\hrule
5566   }
5567 }{
5568   \excludecomment{gnote}
5569 }

```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
5570 \newenvironment{mcb}{
5571   \begin{enumerate}
5572 }{
5573   \end{enumerate}
5574 }
```

we define the keys for the mcc macro

```
5575 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5576   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5577     \bool_set_true:N #1
5578   }{
5579     \bool_set_false:N #1
5580   }
5581 }
5582 \keys_define:nn { problem / mcc }{
5583   id          .str_set_x:N = \l__problems_mcc_id_str ,
5584   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5585   T           .default:n   = { true } ,
5586   T           .bool_set:N  = \l__problems_mcc_t_bool ,
5587   F           .default:n   = { true } ,
5588   F           .bool_set:N  = \l__problems_mcc_f_bool ,
5589   Ttext       .code:n      = {
5590     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5591   } ,
5592   Ftext       .code:n      = {
5593     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5594   }
5595 }
5596 \cs_new_protected:Nn \l__problems_mcc_args:n {
5597   \str_clear:N \l__problems_mcc_id_str
5598   \tl_clear:N \l__problems_mcc_feedback_tl
5599   \bool_set_true:N \l__problems_mcc_t_bool
5600   \bool_set_true:N \l__problems_mcc_f_bool
5601   \bool_set_true:N \l__problems_mcc_Ttext_bool
5602   \bool_set_false:N \l__problems_mcc_Ftext_bool
5603   \keys_set:nn { problem / mcc }{ #1 }
5604 }
```

\mcc

```
5605 \newcommand\mcc[2][] {
5606   \l__problems_mcc_args:n{ #1 }
5607   \item #2
5608   \bool_if:NT \c__problems_solutions_bool {
5609     \
5610     \bool_if:NT \l__problems_mcc_t_bool {
5611       % TODO!
5612       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
5613     }
5614     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

5615         % TODO!
5616         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5617     }
5618     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5619         !
5620     }{
5621         \l__problems_mcc_feedback_tl
5622     }
5623 }
5624 } %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.

```

5625
5626 \keys_define:nn{ problem / inclproblem }{
5627   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5628   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5629   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5630   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5631   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5632   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5633 }
5634 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5635   % \str_clear:N \l__problems_prob_id_str
5636   \tl_clear:N \l__problems_inclprob_pts_tl
5637   \tl_clear:N \l__problems_inclprob_min_tl
5638   \tl_clear:N \l__problems_inclprob_title_tl
5639   \int_zero_new:N \l__problems_inclprob_refnum_int
5640   \str_clear:N \l__problems_inclprob_mhrepos_str
5641   \keys_set:nn { problem / inclproblem }{ #1 }
5642   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5643     \let\l__problems_inclprob_pts_tl\undefined
5644   }
5645   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5646     \let\l__problems_inclprob_min_tl\undefined
5647   }
5648   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5649     \let\l__problems_inclprob_title_tl\undefined
5650   }
5651   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5652     \let\l__problems_inclprob_refnum_int\undefined
5653   }
5654 }
5655
5656 \cs_new_protected:Nn \l__problems_inclprob_clear: {
5657   % \str_clear:N \l__problems_prob_id_str
5658   \let\l__problems_inclprob_pts_tl\undefined
5659   \let\l__problems_inclprob_min_tl\undefined

```

```

5660 \let\l__problems_inclprob_title_tl\undefined
5661 \let\l__problems_inclprob_refnum_int\undefined
5662 \let\l__problems_inclprob_mhrepos_str\undefined
5663 }
5664
5665 \newcommand\includeproblem[2][]{
5666   \__problems_inclprob_args:n{ #1 }
5667   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5668     \input{#2}
5669   }{
5670     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5671       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5672     }
5673   }
5674   \__problems_inclprob_clear:
5675 }

```

(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.

```

5676 \AddToHook{enddocument}{
5677   \bool_if:NT \c__problems_pts_bool {
5678     \message{Total:~\arabic{pts}~points}
5679   }
5680   \bool_if:NT \c__problems_min_bool {
5681     \message{Total:~\arabic{min}~minutes}
5682   }
5683 }

```

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

```

5684 \def\pts#1{
5685   \bool_if:NT \c__problems_pts_bool {
5686     \marginpar{#1~\prob@pt@kw}
5687   }
5688 }
5689 \def\min#1{
5690   \bool_if:NT \c__problems_min_bool {
5691     \marginpar{#1~\prob@min@kw}
5692   }
5693 }

```

\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.

```

5694 \newcounter{pts}
5695 \def\show@pts{
5696   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5697     \bool_if:NT \c__problems_pts_bool {
5698       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5699       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```



```

5700     }
5701   }{
5702     \tl_if_exist:NT \l__problems_prob_pts_tl {
5703       \bool_if:NT \c__problems_pts_bool {
5704         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5705         \addtocounter{pts}{\l__problems_prob_pts_tl}
5706       }
5707     }
5708   }
5709 }

```

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

\show@min

```

5710 \newcounter{min}
5711 \def\show@min{
5712   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5713     \bool_if:NT \c__problems_min_bool {
5714       \marginpar{\l__problems_inclprob_pts_tl;min}
5715       \addtocounter{min}{\l__problems_inclprob_min_tl}
5716     }
5717   }{
5718     \tl_if_exist:NT \l__problems_prob_min_tl {
5719       \bool_if:NT \c__problems_min_bool {
5720         \marginpar{\l__problems_prob_min_tl;min}
5721         \addtocounter{min}{\l__problems_prob_min_tl}
5722       }
5723     }
5724   }
5725 }
5726 \</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.

```
5727 <@@=hwexam>
5728 <*cls>
5729 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5730 \RequirePackage{l3keys2e,expl-keystr-compat}
5731 \DeclareOption*{
5732   \PassOptionsToClass{\CurrentOption}{omdoc}
5733   \PassOptionsToPackage{\CurrentOption}{stex}
5734   \PassOptionsToPackage{\CurrentOption}{hwexam}
5735   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5736 }
5737 \ProcessOptions
```

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

```
5738 \LoadClass{omdoc}
5739 \RequirePackage{stex}
5740 \RequirePackage{hwexam}
5741 \RequirePackage{tikzinput}
5742 \RequirePackage{graphicx}
5743 \RequirePackage{a4wide}
5744 \RequirePackage{amssymb}
5745 \RequirePackage{amstext}
5746 \RequirePackage{amsmath}
```

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

```

5747 \newcommand\assig@default@type{\hwexam@assignment@kw}
5748 \def\document@hwexamtype{\assig@default@type}
5749 <@@=document_structure>
5750 \keys_define:nn { document-structure / document }{
5751 id .str_set_x:N = \c_document_structure_document_id_str,
5752 hwexamtype .tl_set:N = \document@hwexamtype
5753 }
5754 <@@=hwexam>
5755 </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.

```
5756 \*package>
5757 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5758 \RequirePackage{l3keys2e,expl-keystr-compat}
5759
5760 \newif\iftest\testfalse
5761 \DeclareOption{test}{\testtrue}
5762 \newif\ifmultiple\multiplefalse
5763 \DeclareOption{multiple}{\multipletrue}
5764 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5765 \ProcessOptions
```

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

```
5766 \RequirePackage{keyval}[1997/11/10]
5767 \RequirePackage{problem}
```

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

```
5768 \newcommand\hwexam@assignment@kw{Assignment}
5769 \newcommand\hwexam@given@kw{Given}
5770 \newcommand\hwexam@due@kw{Due}
5771 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5772 blank~for~extra~space}%
5773 \newcommand\correction@probs@kw{prob.}%
5774 \newcommand\correction@pts@kw{total}%
5775 \newcommand\correction@reached@kw{reached}%
5776 \newcommand\correction@sum@kw{Sum}%
5777 \newcommand\correction@grade@kw{grade}%
5778 \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

```

5779 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5780
5781 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5782 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5783   \input{hwexam-ngerman.ldf}
5784 }
5785 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5786   \input{hwexam-finnish.ldf}
5787 }
5788 \clist_if_in:NnT \l_tmpa_clist {french}{
5789   \input{hwexam-french.ldf}
5790 }
5791 \clist_if_in:NnT \l_tmpa_clist {russian}{
5792   \input{hwexam-russian.ldf}
5793 }

```

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.

```

5794 \newcounter{assignment}
5795 \numberproblemsin{assignment}
5796 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5797 \keys_define:nn { hwexam / assignment } {
5798   id .str_set:N = \l__hwexam_assign_id_str,
5799   number .int_set:N = \l__hwexam_assign_number_int,
5800   title .tl_set:N = \l__hwexam_assign_title_tl,
5801   type .tl_set:N = \l__hwexam_assign_type_tl,
5802   given .tl_set:N = \l__hwexam_assign_given_tl,
5803   due .tl_set:N = \l__hwexam_assign_due_tl,
5804   loadmodules .code:n = {
5805     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5806   }
5807 }
5808 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5809   \str_clear:N \l__hwexam_assign_id_str
5810   \int_set:Nn \l__hwexam_assign_number_int {-1}
5811   \tl_clear:N \l__hwexam_assign_title_tl
5812   \tl_clear:N \l__hwexam_assign_type_tl
5813   \tl_clear:N \l__hwexam_assign_given_tl
5814   \tl_clear:N \l__hwexam_assign_due_tl
5815   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5816   \keys_set:nn { hwexam / assignment }{ #1 }
5817 }

```

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.

```

5818 \newcommand\given@due[2]{
5819 \bool_lazy_all:nF {
5820 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5821 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5822 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5823 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5824 }{ #1 }
5825
5826 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5827 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5828 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5829 }
5830 }{
5831 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5832 }
5833
5834 \bool_lazy_or:nnF {
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 \bool_lazy_and_p:nn {
5842 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5843 }{
5844 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5845 }
5846 }{ ,~ }
5847
5848 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5849 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5850 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5851 }
5852 }{
5853 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5854 }
5855
5856 \bool_lazy_all:nF {
5857 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5858 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5859 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5860 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5861 }{ #2 }
5862 }

```

`\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.

```

5863 \newcommand\assignment@title[3]{

```

```

5864 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5865 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5866 #1
5867 }{
5868 #2\l__hwexam_assign_title_tl#3
5869 }
5870 }{
5871 #2\l__hwexam_inclassassign_title_tl#3
5872 }
5873 }

```

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

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

```

5874 \newcommand\assignment@number{
5875 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5876 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5877 \int_use:N \l__hwexam_assign_number_int
5878 }
5879 }{
5880 \int_use:N \l__hwexam_inclassassign_number_int
5881 }
5882 }

```

(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.

```

5883 \newenvironment{assignment}[1][]{
5884 \__hwexam_assignment_args:n { #1 }
5885 %\sref@target
5886 \let\__hwexamnum\l__hwexam_assign_number_int
5887 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5888 \stepcounter{assignment}
5889 }{
5890 \setcounter{assignment}{\int_use:N\__hwexamnum}
5891 }
5892 \setcounter{problem}{0}
5893 \def\current@section@level{\document@hwexamtype}
5894 %\sref@label@id{\document@hwexamtype \thesection}
5895 \begin{@assignment}
5896 }{
5897 \end{@assignment}
5898 }

```

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

```

5899 \def\__hwexasstitle{
5900 \protect\document@hwexamtype~\arabic{assignment}
5901 \assignment@title{}\;{} \; -- \given@due{}\}
5902 }

```

```

5903 \ifmultiple
5904 \newenvironment{@assignment}{
5905 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5906 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5907 }{
5908 \begin{omgroup}{\__hwexasstitle}
5909 }
5910 }{
5911 \end{omgroup}
5912 }

```

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

```

5913 \else
5914 \newenvironment{@assignment}{
5915 \begin{center}\bf
5916 \Large\@title\strut\
5917 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\}\}
5918 \large\given@due{--\;}{\;}{--}
5919 \end{center}
5920 }{}
5921 \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.

```

5922 \keys_define:nn { hwexam / inclassignment } {
5923 %id .str_set_x:N = \l__hwexam_assign_id_str,
5924 number .int_set:N = \l__hwexam_inclassign_number_int,
5925 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5926 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5927 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5928 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5929 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5930 }
5931 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5932 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5933 \tl_clear:N \l__hwexam_inclassign_title_tl
5934 \tl_clear:N \l__hwexam_inclassign_type_tl
5935 \tl_clear:N \l__hwexam_inclassign_given_tl
5936 \tl_clear:N \l__hwexam_inclassign_due_tl
5937 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5938 \keys_set:nn { hwexam / inclassignment }{ #1 }
5939 }
5940 \__hwexam_inclassignment_args:n {}
5941
5942 \newcommand\inputassignment[2][ ]{
5943 \__hwexam_inclassignment_args:n { #1 }
5944 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5945 \input{#2}
5946 }{
5947 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```



```

5948 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5949 }
5950 }
5951 \__hwexam_inclasssignment_args:n {}
5952 }
5953 \newcommand\includeassignment[2][ ]{
5954 \newpage
5955 \inputassignment[#1]{#2}
5956 }

```

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

42.4 Typesetting Exams

\quizheading

```

5957 \ExplSyntaxOff
5958 \newcommand\quizheading[1]{%
5959 \def\@tas{#1}%
5960 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5961 \ifx\@tas\empty\else%
5962 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5963 \fi%
5964 }
5965 \ExplSyntaxOn

```

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

\testheading

```

5966 \keys_define:nn { hwexam / testheading } {
5967 min .tl_set:N = \l__hwexam_testheading_min_tl,
5968 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5969 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5970 }
5971 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5972 \tl_clear:N \l__hwexam_testheading_min_tl
5973 \tl_clear:N \l__hwexam_testheading_duration_tl
5974 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5975 \keys_set:nn { hwexam / testheading }{ #1 }
5976 }
5977 \newenvironment{testheading}[1][ ]{
5978 \__hwexam_testheading_args:n{ #1 }
5979 \noindent\large{Name:~\hfill
5980 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5981 \begin{center}
5982 \Large\textbf{\@title}\[1ex]
5983 \large\@date\[3ex]
5984 \end{center}
5985 \textbf{You~have~
5986 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5987 {\l__hwexam_testheading_min_tl}~minutes
5988 }{
5989 {\l__hwexam_testheading_duration_tl}
5990 }~

```

```

5991 (sharp)~for~the~test
5992 };\
5993 Write~the~solutions~to~the~sheet.
5994 \par\noindent
5995 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5996 \advance\check@time by -\theassignment@totalmin
5997 The~estimated~time~for~solving~this~exam~is~
5998 {\theassignment@totalmin}~minutes,~
5999 leaving~you~{\the\check@time}~minutes~for~revising~
6000 your~exam.
6001
6002 \par\noindent
6003 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
6004 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
6005 You~can~reach~{\theassignment@totalpts}~points~if~you~
6006 solve~all~problems.~You~will~only~need~
6007 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
6008 i.e.\ {\the\bonus@pts}~points~are~bonus~points.
6009 \vfill
6010 \begin{center}
6011 {
6012 \Large\em You~have~ample~time,~so~take~it~slow~
6013 and~avoid~rushing~to~mistakes!\}[2ex]
6014 Different~problems~test~different~skills~and~
6015 knowledge,~so~do~not~get~stuck~on~one~problem.
6016 }
6017 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
6018 \end{center}
6019 }{
6020 \newpage
6021 }

```

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

\testspace

```

6022 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6023 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6024 \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.

```

6025 <@=problems>
6026 \renewcommand\@problem[3]{
6027 \stepcounter{assignment@probs}
6028 \def\__problemspts{#2}

```

```

6029 \ifx\__problemspts\@empty\else
6030 \addtocounter{assignment@totalpts}{#2}
6031 \fi
6032 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6033 \xdef\correction@probs{\correction@probs & #1}%
6034 \xdef\correction@pts{\correction@pts & #2}
6035 \xdef\correction@reached{\correction@reached & }
6036 }
6037 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

6038 \newcounter{assignment@probs}
6039 \newcounter{assignment@totalpts}
6040 \newcounter{assignment@totalmin}
6041 \def\correction@probs{\correction@probs@kw}%
6042 \def\correction@pts{\correction@pts@kw}%
6043 \def\correction@reached{\correction@reached@kw}%
6044 \def\after@correction@table{}%
6045 \stepcounter{assignment@probs}
6046 \newcommand\correction@table{
6047 \resizebox{\textwidth}{!}{%
6048 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6049 &\multicolumn{\theassignment@probs}{c|}||%|
6050 {\footnotesize\correction@forgrading@kw} &\\ \hline
6051 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6052 \correction@pts & \theassignment@totalpts & \\ \hline
6053 \correction@reached & & \[.7cm]\hline
6054 \end{tabular}}
6055 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6056 \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\bierrfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\bierrglas{{\bierrfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierrglas}

```