

bbl@beforestart

The sTeX3 Package *

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

2022-02-10

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 2022-02-10)

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	96
28.1	The module environment	99
28.2	Invoking modules	105
29	STeX-Module Inheritance Implementation	107
29.1	SMS Mode	107
29.2	Inheritance	111
30	STeX-Symbols Implementation	116
30.1	Symbol Declarations	116
30.2	Notations	123
31	STeX-Terms Implementation	132
31.1	Symbol Invocations	132
31.2	Terms	135
31.3	Notation Components	141
32	STeX-Structural Features Implementation	144
32.1	Imports with modification	144
32.2	The feature environment	151
32.3	Features	153
33	STeX-Statements Implementation	158
33.1	Definitions	158
33.2	Assertions	161
33.3	Examples	163
33.4	Logical Paragraphs	165

34 The Implementation	168
34.1 Package Options	168
34.2 Proofs	168
34.3 Justifications	174
35 \TeX-Others Implementation	176
36 \TeX-Metatheory Implementation	177
37 Tikzinput Implementation	180
38 document-structure.sty Implementation	182
38.1 The OMDoc Class	182
38.2 Class Options	182
38.3 Beefing up the <code>document</code> environment	183
38.4 Implementation: OMDoc Package	183
38.5 Package Options	183
38.6 Document Structure	185
38.7 Front and Backmatter	188
38.8 Global Variables	190
39 MiKoSlides – Implementation	191
39.1 Class and Package Options	191
39.2 Notes and Slides	193
39.3 Header and Footer Lines	197
39.4 Frame Images	198
39.5 Colors and Highlighting	199
39.6 Sectioning	200
39.7 Excursions	203
40 The Implementation	204
40.1 Package Options	204
40.2 Problems and Solutions	205
40.3 Multiple Choice Blocks	210
40.4 Including Problems	211
40.5 Reporting Metadata	212
41 Implementation: The hwexam Class	214
41.1 Class Options	214
42 Implementation: The hwexam Package	216
42.1 Package Options	216
42.2 Assignments	217
42.3 Including Assignments	220
42.4 Typesetting Exams	221
42.5 Leftovers	223

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

- **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/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `source-base` or
`ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),
- `narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),
- `url`: The URL that is formed as a basis for *external references*, see (TODO),
- `dependencies`: All archives that this archive depends on. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

Chapter 5

Creating New Modules and Symbols

TODO

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

Example 3

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

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

Module 5.1.5[int]

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 4

```
\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 5

```
\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 6

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

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

.

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

Example 7

```
\mult[\comp{Multiplying}]* $a$  $b$  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 8

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

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 9

```
\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 10

```
\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 11

```
\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 12

```
\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 13

```
\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⟩}`

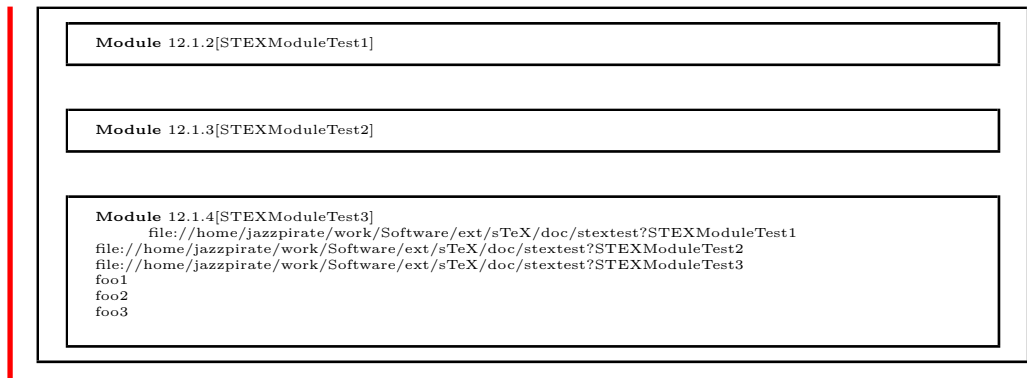
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\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

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 14.1.2[NotationTest]

`\symdef`

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

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

Test 13

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

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

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

<hr/> <hr/> <code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
<hr/> <hr/> <code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
<hr/> <hr/> <code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . In math mode, it uses the default or optionally provided notation of the associated symbol. If followed by <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code>), i.e. it allows to refer to <code>\plus!</code> [addition] as an operation, rather than <code>\plus[addition of]{some}{terms}</code> .
<hr/> <hr/> <code>_stex_term_math_oms:nnnn</code> <code>_stex_term_math_oma:nnnn</code> <code>_stex_term_math_omb:nnnn</code>	<code><URI><fragment><precedence><body></code> Annotates <code><body></code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code><URI></code> , generated by the specific notation <code><fragment></code> with (upwards) operator precedence <code><precedence></code> . Inserts parentheses according to the current downwards precedence and operator precedence.
<hr/> <hr/> <code>_stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn<int><prec><body></code> Annotates <code><body></code> as the <code><int></code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> .
<hr/> <hr/> <code>_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn<int><prec><notation><body></code> Annotates <code><body></code> as the <code><int></code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> and associative notation <code><notation></code> .

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

Test 14

```

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

```

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

Test 15

```

\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle } }
$\bar{foo} a\{b,c,d,e,f\}g$ 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 $\mult{a,\plus{\frac{ab}{ac}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{ac}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle \mult{a,\plus{\frac{ab}{ac}}}$}
\end{module}

```

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

`\stex_term_custom:nn`

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

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

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

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 16

TeX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

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

21.1 Introduction

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

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

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

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

omgroup

id

creators

contributors

short

loadmodules

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 `LATEXML` transformation.

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

```

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

```

\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

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy`[$\langle URL \rangle$]{ $\langle label \rangle$ }, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL $\langle URL \rangle$ that lets \LaTeX ML generate the correct reference.

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.¹⁰

21.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the \LaTeX preamble of the course notes file. `\setSGvar`{ $\langle vname \rangle$ }{ $\langle text \rangle$ } to set the global variable $\langle vname \rangle$ to $\langle text \rangle$ and `\useSGvar`{ $\langle vname \rangle$ } to reference it.

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

The `mikoslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the \TeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

22.2.1 Package Options

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

- | | |
|---------------------------|---|
| <code>slides</code> | • 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.

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

Name:

320101 General Computer Science (Fall 2010)

2022-02-10

You have 60 minutes (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

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

[illegible]

good luck

Example 8: A generated test heading.

Part IV
Implementation

Chapter 25

STEX -Basics Implementation

25.1 The STEXDocument 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:nxx{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:N \underline{TF}`

```

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:Nn \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   \stex_get_document_uri:
878   \stex_if_smsmode:TF {
879     \stex_get_document_url:
880     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
881     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
882   }{
883     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel\iffalse}}
884     \exp_args:Nx\label{sref_sym_#1}
885
886     \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
887     \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
888   }
889 }
890 }

```

27.3 Using References

```

891 \str_new:N \l__stex_refs_indocument_str
892 \keys_define:nn { stex / sref } {
893   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
894   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
895   pre           .tl_set:N = \l__stex_refs_pre_tl ,
896   post          .tl_set:N = \l__stex_refs_post_tl ,
897   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
898 }
899
900 \bool_new:N \c__stex_refs_hyperref_bool
901 \bool_set_false:N \c__stex_refs_hyperref_bool
902 \AddToHook{begindocument}{
903   \ifpackageloaded{hyperref}{
904     \bool_set_true:N \c__stex_refs_hyperref_bool
905   }{}
906 }
907

```

```

908
909 \cs_new_protected:Nn \__stex_refs_args:n {
910   \tl_clear:N \l__stex_refs_linktext_tl
911   \tl_clear:N \l__stex_refs_fallback_tl
912   \tl_clear:N \l__stex_refs_pre_tl
913   \tl_clear:N \l__stex_refs_post_tl
914   \str_clear:N \l__stex_refs_repo_str
915   \keys_set:nn { stex / sref } { #1 }
916 }
917
918 \NewDocumentCommand \sref { 0{} m}{
919   \__stex_refs_args:n { #1 }
920   \str_if_empty:NTF \l__stex_refs_indocument_str {
921     \str_set:Nn \l_tmpa_str { #2 }
922     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
923     \tl_set:Nn \l_tmpa_tl {
924       \l__stex_refs_fallback_tl
925     }
926     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
927       \str_set:Nn \l_tmpb_str { ##1 }
928       \str_if_eq:eeT { \l_tmpa_str } {
929         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
930       } {
931         \seq_map_break:n {
932           \tl_set:Nn \l_tmpa_tl {
933             % doc uri in \l_tmpb_str
934             \str_set:Nx \l_tmpa_str {\use:c{sref\l_tmpb_str_type}}
935             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
936               % reference
937               \cs_if_exist:cTF{autoref}{
938                 \l__stex_refs_pre_tl\autoref{sref\l_tmpb_str}\l__stex_refs_post_tl
939               }{
940                 \l__stex_refs_pre_tl\ref{sref\l_tmpb_str}\l__stex_refs_post_tl
941               }
942             }{
943               % URL
944               \if_bool:N \c__stex_refs_hyperref_bool {
945                 \exp_args:Nx \href{\use:c{sref_url\l_tmpb_str_str}}{\l__stex_refs_fallback
946               }{
947                 \l__stex_refs_fallback_tl
948               }
949             }
950           }
951         }
952       }
953     }
954     \l_tmpa_tl
955   }{
956     % TODO
957   }
958 }
959
960 \NewDocumentCommand \srefsym { 0{} m}{
961   \stex_get_symbol:n { #2 }

```

```

962 \__stex_refs_args:n { #1 }
963 \str_if_empty:NTF \l__stex_refs_indocument_str {
964   \tl_set:Nn \l_tmpa_tl {
965     \l__stex_refs_fallback_tl
966   }
967   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str _type}{
968     \tl_set:Nn \l_tmpa_tl {
969       % doc uri in \l_tmpb_str
970       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str _type}}
971       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
972         % reference
973         \cs_if_exist:cTF{autoref}{
974           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
975         }{
976           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
977         }
978       }{
979         % URL
980         \if_bool:N \c__stex_refs_hyperref_bool {
981           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str _str}}{\l__stex_refs_fallback_tl}
982         }{
983           \l__stex_refs_fallback_tl
984         }
985       }
986     }
987   }
988   \l_tmpa_tl
989 }{
990   % TODO
991 }
992 }
993
994 \cs_new_protected:Npn \srefsymuri #1 #2 {
995   \hyperref[sref_sym_#1]{#2}
996 }
997
998 </package>

```

Chapter 28

STEX -Modules Implementation

```
999 <*package>
1000
1001 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1002
1003 <@@=stex_modules>
1004
1005     Warnings and error messages
1006 \msg_new:nnn{stex}{error/unknownmodule}{
1007     No~module~#1~found
1008 }
1009 \msg_new:nnn{stex}{error/syntax}{
1010     Syntax~error:~#1
1011 }
1012 \msg_new:nnn{stex}{error/siglanguage}{
1013     Module~#1~declares~signature~#2,~but~does~not~
1014     declare~its~language
1015 }
1016 \msg_new:nnn{stex}{error/concllictingmodules}{
1017     Conflicting~imports~for~module~#1
1018 }
1019
1020 \l_stex_current_module_str The current module:
1021 \str_new:N \l_stex_current_module_str
1022
1023 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1024
1025 \l_stex_all_modules_seq Stores all available modules
1026 \seq_new:N \l_stex_all_modules_seq
1027
1028 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1029
1030 \stex_if_in_module_p:
1031 \stex_if_in_module:TF
1032 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1033     \str_if_empty:NTF \l_stex_current_module_str
1034     \prg_return_false: \prg_return_true:
1035 }
```

(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`

```
1024 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1025   \prop_if_exist:cTF { c_stex_module_#1_prop }
1026   \prg_return_true: \prg_return_false:
1027 }
```

(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:

```
1028 \cs_new_protected:Nn \stex_add_to_current_module:n {
1029   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1030 }
1031 \cs_new_protected:Npn \STEXexport {
1032   \begingroup
1033   \newlinechar=-1\relax
1034   \endlinechar=-1\relax
1035   %\catcode'\ = 9\relax
1036   \expandafter\endgroup\STEXexport:n
1037 }
1038 \cs_new_protected:Nn \STEXexport:n {
1039   \ignorespaces #1
1040   \stex_add_to_current_module:n { \ignorespaces #1 }
1041   \stex_smsmode_set_codes:
1042 }
1043 \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`

```
1044 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1045   \str_set:Nx \l_tmpa_str { #1 }
1046   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1047 }
1048
1049 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1050 % \str_set:Nx \l_tmpa_str { #1 }
1051 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1052 %}
```

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

`\stex_collect_imports:n`

```
1053 \cs_new_protected:Nn \stex_collect_imports:n {
1054   \seq_clear:N \l_stex_collect_imports_seq
1055   \__stex_modules_collect_imports:n {#1}
1056 }
1057 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1058   \seq_map_inline:cn {c_stex_module_#1_imports} {
1059     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1060       \__stex_modules_collect_imports:n { ##1 }
1061     }
1062 }
```



```

1062 }
1063 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1064   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1065 }
1066 }

```

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

`\stex_add_import_to_current_module:n`

```

1067 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1068   \str_set:Nx \l_tmpa_str { #1 }
1069   \exp_args:Nno
1070   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1071     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1072   }
1073 }

```

(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).

```

1074 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1075   \str_set:Nx \l_tmpa_str { #1 }
1076   \seq_set_eq:NN \l_tmpa_seq #2
1077   % split off file extension
1078   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1079   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1080   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1081   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1082
1083   \bool_set_true:N \l_tmpa_bool
1084   \bool_while_do:Nn \l_tmpa_bool {
1085     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1086     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1087       {source} { \bool_set_false:N \l_tmpa_bool }
1088     }{}{
1089       \seq_if_empty:NT \l_tmpa_seq {
1090         \bool_set_false:N \l_tmpa_bool
1091       }
1092     }
1093   }
1094
1095   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1096   \str_if_empty:NTF \l_stex_modules_subpath_str {
1097     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1098   }{
1099     \str_set:Nx \l_stex_modules_ns_str {
1100       \l_tmpa_str/\l_stex_modules_subpath_str
1101     }
1102   }
1103 }

```

(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
1104 \str_new:N \l_stex_modules_ns_str
1105 \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.

```

1106 \cs_new_protected:Nn \stex_modules_current_namespace: {
1107   \str_clear:N \l_stex_modules_subpath_str
1108   \prop_if_exist:NTF \l_stex_current_repository_prop {
1109     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1110     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1111   }{
1112     % split off file extension
1113     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1114     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1115     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1116     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1117     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1118     \str_set:Nx \l_stex_modules_ns_str {
1119       file:/\stex_path_to_string:N \l_tmpa_seq
1120     }
1121   }
1122 }

```

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

28.1 The module environment

module arguments:

```

1123 \keys_define:nn { stex / module } {
1124   title      .str_set_x:N = \l_stex_module_title_str ,
1125   ns         .str_set_x:N = \l_stex_module_ns_str ,
1126   lang       .str_set_x:N = \l_stex_module_lang_str ,
1127   sig        .str_set_x:N = \l_stex_module_sig_str ,
1128   creators   .str_set_x:N = \l_stex_module_creators_str ,
1129   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1130   meta       .str_set_x:N = \l_stex_module_meta_str ,
1131   srccite    .str_set_x:N = \l_stex_module_srccite_str
1132 }
1133
1134 \cs_new_protected:Nn \__stex_modules_args:n {
1135   \str_clear:N \l_stex_module_title_str
1136   \str_clear:N \l_stex_module_ns_str
1137   \str_clear:N \l_stex_module_lang_str
1138   \str_clear:N \l_stex_module_sig_str
1139   \str_clear:N \l_stex_module_creators_str
1140   \str_clear:N \l_stex_module_contributors_str
1141   \str_clear:N \l_stex_module_meta_str
1142   \str_clear:N \l_stex_module_srccite_str
1143   \keys_set:nn { stex / module } { #1 }

```

```

1144 }
1145
1146 % module parameters here? In the body?
1147

```

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

```

1148 \cs_new_protected:Nn \stex_module_setup:nn {
1149   \str_set:Nx \l_stex_module_name_str { #2 }
1150   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1151 \stex_if_in_module:TF {
1152   % Nested module
1153   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1154   { ns } \l_stex_module_ns_str
1155   \str_set:Nx \l_stex_module_name_str {
1156     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1157     { name } / \l_stex_module_name_str
1158   }
1159 }{
1160   % not nested:
1161   \str_if_empty:NT \l_stex_module_ns_str {
1162     \stex_modules_current_namespace:
1163     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1164     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1165       / {\l_stex_module_ns_str}
1166     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1167     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1168       \str_set:Nx \l_stex_module_ns_str {
1169         \stex_path_to_string:N \l_tmpa_seq
1170       }
1171     }
1172   }
1173 }

```

Next, we determine the language of the module:

```

1174 \str_if_empty:NT \l_stex_module_lang_str {
1175   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1176   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1177   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1178   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1179   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1180     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1181       inferred~from~file~name}
1182     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1183   }
1184 }
1185
1186 \str_if_empty:NF \l_stex_module_lang_str {
1187   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1188   \l_tmpa_str {
1189     \ltx@ifpackageloaded{babel}{
1190       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1191     }{}
1192   } {
1193     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1194   }
1195 }

```

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

```

1196 \str_if_empty:NTF \l_stex_module_sig_str {
1197   \exp_args:Nnx \prop_gset_from_keyval:cn {
1198     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1199   } {
1200     name      = \l_stex_module_name_str ,
1201     ns        = \l_stex_module_ns_str ,
1202     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1203     lang      = \l_stex_module_lang_str ,
1204     sig       = \l_stex_module_sig_str ,
1205     meta      = \l_stex_module_meta_str
1206   }
1207   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1208   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1209   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1210   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1211   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1212 \str_if_empty:NT \l_stex_module_meta_str {
1213   \str_set:Nx \l_stex_module_meta_str {
1214     \c_stex_metatheory_ns_str ? Metatheory
1215   }
1216 }
1217 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1218   \bool_set_true:N \l_stex_in_meta_bool
1219   \exp_args:Nx \stex_add_to_current_module:n {
1220     \bool_set_true:N \l_stex_in_meta_bool
1221     \stex_activate_module:n {\l_stex_module_meta_str}
1222     \bool_set_false:N \l_stex_in_meta_bool
1223   }
1224   \stex_activate_module:n {\l_stex_module_meta_str}
1225   \bool_set_false:N \l_stex_in_meta_bool
1226 }
1227 }{
1228   \str_if_empty:NT \l_stex_module_lang_str {
1229     \msg_error:nnxx{stex}{error/siglanguage}{
1230       \l_stex_module_ns_str?\l_stex_module_name_str
1231     }{\l_stex_module_sig_str}
1232   }
1233
1234   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1235   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1236   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1237   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1238   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1239   \str_set:Nx \l_tmpa_str {

```

```

1240     \stex_path_to_string:N \l_tmpa_seq /
1241     \l_tmpa_str . \l_stex_module_sig_str .tex
1242   }
1243   \IfFileExists \l_tmpa_str {
1244     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1245       \seq_clear:N \l_stex_all_modules_seq
1246       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1247       \input { \l_tmpa_str }
1248     }
1249   }{
1250     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1251   }
1252   \stex_if_smsmode:F {
1253     \stex_activate_module:n {
1254       \l_stex_module_ns_str ? \l_stex_module_name_str
1255     }
1256   }
1257   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1258 }
1259 }

```

(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}

1260 \int_new:N \l_stex_module_group_depth_int
1261 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1262   \stex_reactivate_macro:N \STEXexport
1263   \stex_reactivate_macro:N \importmodule
1264   \stex_reactivate_macro:N \symdecl
1265   \stex_reactivate_macro:N \notation
1266   \stex_reactivate_macro:N \symdef
1267   \stex_module_setup:nn{#1}{#2}
1268 }
1269 \stex_debug:nn{modules}{
1270   New~module:\\
1271   Namespace:~\l_stex_module_ns_str\\
1272   Name:~\l_stex_module_name_str\\
1273   Language:~\l_stex_module_lang_str\\
1274   Signature:~\l_stex_module_sig_str\\
1275   Metatheory:~\l_stex_module_meta_str\\
1276   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1277 }
1278
1279 \seq_put_right:Nx \l_stex_all_modules_seq {
1280   \l_stex_module_ns_str ? \l_stex_module_name_str
1281 }
1282
1283 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1284 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1285
1286
1287 \stex_if_smsmode:TF {

```

```

1288 \stex_smsmode_set_codes:
1289 } {
1290 \begin{stex_annotate_env} {theory} {
1291 \l_stex_module_ns_str ? \l_stex_module_name_str
1292 }
1293
1294 \stex_annotate_invisible:nnn{header}{} {
1295 \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1296 \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1297 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1298 \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1299 }
1300 }
1301 }
1302 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1303 % TODO: Inherit metatheory for nested modules?
1304 }
1305 \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}

```

1306 \cs_new_protected:Nn \_stex_modules_end_module: {
1307 % \str_set:Nx \l_tmpa_str {
1308 % c_stex_module_
1309 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1310 % \prop_item:Nn \l_stex_current_module_prop { name }
1311 % _prop
1312 % }
1313 %^^A \prop_new:c { \l_tmpa_str }
1314 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1315 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1316 }

```

(End definition for _stex_modules_end_module:.)

@module The core environment, with no header

```

1317 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1318 \NewDocumentEnvironment { @module } { 0 } { m } {
1319 \par
1320 \_stex_modules_begin_module:nn{#1}{#2}
1321 } {
1322 \_stex_modules_end_module:
1323 \stex_if_smsmode:TF {
1324 % \exp_args:Nx \stex_add_to_sms:n {
1325 % \prop_gset_from_keyval:cn {
1326 % c_stex_module_
1327 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1328 % \prop_item:Nn \l_stex_current_module_prop { name }
1329 % _prop
1330 % } {
1331 % name = \prop_item:cn { \l_tmpa_str } { name } ,
1332 % ns = \prop_item:cn { \l_tmpa_str } { ns } ,
1333 % file = \prop_item:cn { \l_tmpa_str } { file } ,

```

```

1334 %      lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1335 %      sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1336 %      meta      = \prop_item:cn { \l_tmpa_str } { meta }
1337 %    }
1338 %  }
1339 }{
1340   \end{stex_annotate_env}
1341 }
1342 }

```

\stex_modules_heading: Code for document headers

```

1343 \cs_if_exist:NTF \thesection {
1344   \newcounter{module}[section]
1345 }{
1346   \newcounter{module}
1347 }
1348
1349 \bool_if:NT \c_stex_showmods_bool {
1350   \latexml_if:F { \RequirePackage{mdframed} }
1351 }
1352
1353 \cs_new_protected:Nn \stex_modules_heading: {
1354   \stepcounter{module}
1355   \par
1356   \bool_if:NT \c_stex_showmods_bool {
1357     \noindent{\textbf{Module} ~
1358       \cs_if_exist:NT \thesection {\thesection.}
1359       \themodule ~ [\l_stex_module_name_str]
1360     }
1361     \str_if_empty:NTF \l_stex_module_title_str {
1362       }{
1363         \quad(\l_stex_module_title_str)\hfill
1364       }\par
1365     }
1366     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1367     % TODO
1368     \stex_ref_new_doc_target:n \l_stex_module_name_str
1369   }

```

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

Finally:

```

1370 \NewDocumentEnvironment { module } { 0 } { m } {
1371   \bool_if:NT \c_stex_showmods_bool {
1372     \begin{mdframed}
1373   }
1374   \begin{@module}[#1]{#2}
1375     \stex_modules_heading:
1376   }{
1377     \end{@module}
1378     \bool_if:NT \c_stex_showmods_bool {
1379       \end{mdframed}
1380     }
1381   }

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1382 \NewDocumentCommand \STEXModule { m } {
1383   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1384   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1385   \tl_set:Nn \l_tmpa_tl {
1386     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1387   }
1388   \seq_map_inline:Nn \l_stex_all_modules_seq {
1389     \str_set:Nn \l_tmpb_str { ##1 }
1390     \str_if_eq:eeT { \l_tmpa_str } {
1391       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1392     } {
1393       \seq_map_break:n {
1394         \tl_set:Nn \l_tmpa_tl {
1395           \stex_invoke_module:n { ##1 }
1396         }
1397       }
1398     }
1399   }
1400   \l_tmpa_tl
1401 }
1402
1403 \cs_new_protected:Nn \stex_invoke_module:n {
1404   \stex_debug:nn{modules}{Invoking~module~#1}
1405   \peek_charcode_remove:NTF ! {
1406     \__stex_modules_invoke_uri:nN { #1 }
1407   } {
1408     \peek_charcode_remove:NTF ? {
1409       \__stex_modules_invoke_symbol:nn { #1 }
1410     } {
1411       \msg_error:nnx{stex}{error/syntax}{
1412         ?~or~!~expected~after~
1413         \c_backslash_str STEXModule{#1}
1414       }
1415     }
1416   }
1417 }
1418
1419 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1420   \str_set:Nn #2 { #1 }
1421 }
1422
1423 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1424   \stex_invoke_symbol:n{#1?#2}
1425 }

```

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

```

\stex_activate_module:n
1426 \bool_new:N \l_stex_in_meta_bool
1427 \bool_set_false:N \l_stex_in_meta_bool

```



```

1428 \cs_new_protected:Nn \stex_activate_module:n {
1429   \stex_debug:nn{modules}{Activating~module~#1}
1430   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1431     \msg_error:nnn{stex}{error/concllictingmodules}{ #1 }
1432   }
1433   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1434     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1435     \use:c{ c_stex_module_#1_code }
1436   }
1437 }

```

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

```

1438 </package>

```

Chapter 29

sTeX -Module Inheritance Implementation

```
1439 ⟨*package⟩
1440
1441 %%%%%%%%% inheritance.dtx %%%%%%%%%
1442
```

29.1 SMS Mode

```
1443 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1444 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1445 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1446 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1447
1448 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1449   \makeatletter
1450   \makeatother
1451   \ExplSyntaxOn
1452   \ExplSyntaxOff
1453 }
1454
1455 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1456   \symdef
1457   \importmodule
1458   \notation
1459   \symdecl
1460   \STEXexport
1461 }
1462
1463 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1464   \tl_to_str:n {
1465     module,
1466     @module
```

```

1467 }
1468 }

```

(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

```

```

1469 \bool_new:N \g__stex_smsmode_bool
1470 \bool_set_false:N \g__stex_smsmode_bool
1471 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1472   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1473 }

```

(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

```

```

1474 \bool_new:N \g__stex_smsmode_catcode_bool
1475 \bool_set_false:N \g__stex_smsmode_catcode_bool
1476 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1477   \bool_if:NTF \g__stex_smsmode_catcode_bool
1478   \prg_return_true: \prg_return_false:
1479 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1480 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1481   \stex_if_smsmode:T {
1482     \__stex_smsmode_if_catcodes:F {
1483       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1484       \exp_after:wN \char_gset_active_eq:NN
1485       \c_backslash_str \__stex_smsmode_cs:
1486       \tex_global:D \char_set_catcode_active:N \
1487       \tex_global:D \char_set_catcode_other:N $
1488       \tex_global:D \char_set_catcode_other:N ^
1489       \tex_global:D \char_set_catcode_other:N _
1490       \tex_global:D \char_set_catcode_other:N &
1491       \tex_global:D \char_set_catcode_other:N ##
1492     }
1493   }
1494 } \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.

```

1495 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1496   \__stex_smsmode_if_catcodes:T {
1497     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1498     \exp_after:wN \tex_global:D \exp_after:wN
1499     \char_set_catcode_escape:N \c_backslash_str
1500     \tex_global:D \char_set_catcode_math_toggle:N $
1501     \tex_global:D \char_set_catcode_math_superscript:N ^
1502     \tex_global:D \char_set_catcode_math_subscript:N _
1503     \tex_global:D \char_set_catcode_alignment:N &
1504     \tex_global:D \char_set_catcode_parameter:N ##
1505   }
1506 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1507 \cs_new_protected:Nn \stex_in_smsmode:nn {
1508   \vbox_set:Nn \l_tmpa_box {
1509     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1510     \bool_gset_true:N \g__stex_smsmode_bool
1511     \stex_smsmode_set_codes:
1512     #2
1513     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1514     \stex_if_smsmode:F {
1515       \__stex_smsmode_unset_codes:
1516     }
1517   }
1518   \box_clear:N \l_tmpa_box
1519 }

```

(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:

```

1520 \cs_new_protected:Nn \_stex_smsmode_cs: {
1521   \str_clear:N \l_tmpa_str
1522   \peek_analysis_map_inline:n {
1523     % #1: token (one expansion)
1524     % #2: charcode
1525     % #3 catcode
1526     \token_if_eq_charcode:NNTF ##3 B {
1527       % token is a letter
1528       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1529     } {
1530       \str_if_empty:NTF \l_tmpa_str {
1531         % we don't allow (or need) single non-letter CSs
1532         % for now
1533         \peek_analysis_map_break:
1534       }{
1535         \str_if_eq:onTF \l_tmpa_str { begin } {
1536           \peek_analysis_map_break:n {
1537             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1538           }
1539         } {
1540           \str_if_eq:onTF \l_tmpa_str { end } {
1541             \peek_analysis_map_break:n {
1542               \exp_after:wN \_stex_smsmode_checkend:n ##1
1543             }
1544           } {
1545             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1546             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1547             \g_stex_smsmode_allowedmacros_tl
1548             { \use:c{\l_tmpa_str} } {
1549               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1550               \peek_analysis_map_break:n {
1551                 \exp_after:wN \l_tmpa_tl ##1
1552               }

```

```

1553     } {
1554         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1555         \g_stex_smsmode_allowedmacros_escape_tl
1556         { \use:c{\l_tmpa_str} } {
1557             \__stex_smsmode_unset_codes:
1558             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1559             % TODO \__stex_smsmode_rescan_cs:
1560             % \int_compare:nNnTF {##2} = {92} {
1561             %     \peek_analysis_map_break:n {
1562             %         \__stex_smsmode_unset_codes:
1563             %         \__stex_smsmode_rescan_cs:
1564             %     }
1565             % } {
1566             %     \peek_analysis_map_break:n {
1567             %         \exp_after:wN \l_tmpa_tl ##1
1568             %     }
1569             % }
1570         } {
1571             \int_compare:nNnTF {##2} = {92} {
1572                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1573             }{
1574                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1575             }
1576         }
1577     }
1578 }
1579 }
1580 }
1581 }
1582 }
1583 }

```

(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:

```

1584 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1585     \str_clear:N \l_tmpb_str
1586     \peek_analysis_map_inline:n {
1587         \token_if_eq_charcode:NNTF ##3 B {
1588             % token is a letter
1589             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1590         } {
1591             \peek_analysis_map_break:n {
1592                 \exp_after:wN \use:c \exp_after:wN {
1593                     \exp_after:wN \l_tmpa_str\exp_after:wN
1594                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1595             }
1596         }
1597     }
1598 }

```

(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.

```

1599 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1600   \str_set:Nn \l_tmpa_str { #1 }
1601   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1602     \__stex_smsmode_unset_codes:
1603     \begin{#1}
1604   }
1605 }
```

(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.

```

1606 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1607   \str_set:Nn \l_tmpa_str { #1 }
1608   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1609     \end{#1}
1610   }
1611 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1612 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1613 \cs_new_protected:Nn \stex_import_module_uri:nn {
1614   \str_set:Nx \l_stex_import_archive_str { #1 }
1615   \str_set:Nn \l_stex_import_path_str { #2 }
1616
1617   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1618   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1619   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1620
1621   \stex_modules_current_namespace:
1622   \bool_lazy_all:nTF {
1623     {\str_if_empty_p:N \l_stex_import_archive_str}
1624     {\str_if_empty_p:N \l_stex_import_path_str}
1625     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1626   }{
1627     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1628     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1629   }{
1630     \str_if_empty:NT \l_stex_import_archive_str {
1631       \prop_if_exist:NT \l_stex_current_repository_prop {
1632         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1633       }
1634     }
1635     \str_if_empty:NTF \l_stex_import_archive_str {
1636       \str_if_empty:NF \l_stex_import_path_str {
1637         \str_set:Nx \l_stex_import_ns_str {
1638           \l_stex_module_ns_str / \l_stex_import_path_str
1639         }
1640       }
1641     }
```

```

1641   }{
1642     \stex_require_repository:n \l_stex_import_archive_str
1643     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1644     \l_stex_import_ns_str
1645     \str_if_empty:NF \l_stex_import_path_str {
1646       \str_set:Nx \l_stex_import_ns_str {
1647         \l_stex_import_ns_str / \l_stex_import_path_str
1648       }
1649     }
1650   }
1651 }
1652 }

```

(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 1653 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1654 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1655 \str_new:N \l_stex_import_path_str
                            1656 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1657 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1658   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1659
1660     % archive
1661     \str_set:Nx \l_tmpa_str { #2 }
1662     \str_if_empty:NTF \l_tmpa_str {
1663       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1664     } {
1665       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1666       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1667       \seq_put_right:Nn \l_tmpa_seq { source }
1668     }
1669
1670     % path
1671     \str_set:Nx \l_tmpb_str { #3 }
1672     \str_if_empty:NTF \l_tmpb_str {
1673       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1674
1675       \ltx@ifpackageloaded{babel} {
1676         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1677           { \language } \l_tmpb_str {
1678           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1679         }
1680       } {
1681         \str_clear:N \l_tmpb_str
1682       }
1683
1684       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1685       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1686         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1687     }{
1688       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1689       \IfFileExists{ \l_tmpa_str.tex }{
1690         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1691       }{
1692         % try english as default
1693         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1694         \IfFileExists{ \l_tmpa_str.en.tex }{
1695           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1696         }{
1697           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1698         }
1699       }
1700     }
1701
1702   } {
1703     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1704     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1705
1706     \ltx@ifpackageloaded{babel} {
1707       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1708       { \language } \l_tmpb_str {
1709         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1710       }
1711     } {
1712       \str_clear:N \l_tmpb_str
1713     }
1714
1715     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1716
1717     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1718     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1719       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1720     }{
1721       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1722       \IfFileExists{ \l_tmpa_str/#4.tex }{
1723         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1724       }{
1725         % try english as default
1726         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1727         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1728           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1729         }{
1730           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1731           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1732             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1733           }{
1734             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1735             \IfFileExists{ \l_tmpa_str.tex }{
1736               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1737             }{
1738               % try english as default
1739               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1740               \IfFileExists{ \l_tmpa_str.en.tex }{

```



```

1741         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1742     }{
1743         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1744     }
1745 }
1746 }
1747 }
1748 }
1749 }
1750 }
1751
1752 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1753     \seq_clear:N \l_stex_all_modules_seq
1754     \str_clear:N \l_stex_current_module_str
1755     \str_set:Nx \l_tmpb_str { #2 }
1756     \str_if_empty:NF \l_tmpb_str {
1757         \stex_set_current_repository:n { #2 }
1758     }
1759     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1760     \input { \g__stex_importmodule_file_str }
1761 }
1762
1763 \stex_if_module_exists:nF { #1 ? #4 } {
1764     \msg_error:nnx{stex}{error/unknownmodule}{
1765         #1?#4~(in~file~\g__stex_importmodule_file_str)
1766     }
1767 }
1768 }
1769 \stex_activate_module:n { #1 ? #4 }
1770 }

```

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

`\importmodule`

```

1771 \NewDocumentCommand \importmodule { 0{} m } {
1772     \stex_import_module_uri:nn { #1 } { #2 }
1773     \stex_debug:nn{modules}{Importing~module:~
1774         \l_stex_import_ns_str ? \l_stex_import_name_str
1775     }
1776     \stex_if_smsmode:F {
1777         \stex_import_require_module:nnnn
1778         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1779         { \l_stex_import_path_str } { \l_stex_import_name_str }
1780         \stex_annotate_invisible:nnn
1781         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1782     }
1783     \exp_args:Nx \stex_add_to_current_module:n {
1784         \stex_import_require_module:nnnn
1785         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1786         { \l_stex_import_path_str } { \l_stex_import_name_str }
1787     }
1788     \exp_args:Nx \stex_add_import_to_current_module:n {
1789         \l_stex_import_ns_str ? \l_stex_import_name_str
1790     }

```

```

1791 \stex_smsmode_set_codes:
1792 }
1793 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1794 \NewDocumentCommand \usemodule { 0{} m } {
1795   \stex_if_smsmode:F {
1796     \stex_import_module_uri:nn { #1 } { #2 }
1797     \stex_import_require_module:nnnn
1798     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1799     { \l_stex_import_path_str } { \l_stex_import_name_str }
1800     \stex_annotate_invisible:nnn
1801     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1802   }
1803   \stex_smsmode_set_codes:
1804 }

```

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

```

1805 \endpackage

```

Chapter 30

STEX -Symbols Implementation

```
1806 <*package>
1807
1808 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1809
Warnings and error messages
1810
```

30.1 Symbol Declarations

```
1811 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1812 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol
1813 \NewDocumentCommand \STEXsymbol { m } {
1814   \stex_get_symbol:n { #1 }
1815   \exp_args:No
1816   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1817 }

(End definition for \STEXsymbol. This function is documented on page 38.)
symdecl arguments:
1818 \keys_define:nn { stex / symdecl } {
1819   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1820   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1821   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1822   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1823   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1824   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1825   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1826   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1827 }
```

```

1828
1829 \bool_new:N \l_stex_symdecl_make_macro_bool
1830
1831 \cs_new_protected:Nn \__stex_symdecl_args:n {
1832   \str_clear:N \l_stex_symdecl_name_str
1833   \str_clear:N \l_stex_symdecl_args_str
1834   \bool_set_false:N \l_stex_symdecl_local_bool
1835   \tl_clear:N \l_stex_symdecl_type_tl
1836   \tl_clear:N \l_stex_symdecl_definiens_tl
1837
1838   \keys_set:nn { stex / symdecl } { #1 }
1839 }

```

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

```

1840
1841 \NewDocumentCommand \symdecl { s O{} m } {
1842   \__stex_symdecl_args:n { #2 }
1843   \IfBooleanTF #1 {
1844     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1845   } {
1846     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1847   }
1848   \stex_symdecl_do:n { #3 }
1849   \stex_smsmode_set_codes:
1850 }
1851 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1852 \cs_new_protected:Nn \stex_symdecl_do:n {
1853   \stex_if_in_module:F {
1854     % TODO throw error? some default namespace?
1855   }
1856
1857   \str_if_empty:NT \l_stex_symdecl_name_str {
1858     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1859   }
1860
1861   \prop_if_exist:cT { l_stex_symdecl_
1862     \l_stex_current_module_str ?
1863     \l_stex_symdecl_name_str
1864     _prop
1865   }{
1866     % TODO throw error (beware of circular dependencies)
1867   }
1868
1869   \prop_clear:N \l_tmpa_prop
1870   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1871   \seq_clear:N \l_tmpa_seq
1872   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1873   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1874

```

```

1875 \exp_args:No \stex_add_constant_to_current_module:n {
1876   \l_stex_symdecl_name_str
1877 }
1878
1879 % arity/args
1880 \int_zero:N \l_tmpb_int
1881
1882 \bool_set_true:N \l_tmpa_bool
1883 \str_map_inline:Nn \l_stex_symdecl_args_str {
1884   \token_case_meaning:NnF ##1 {
1885     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1886     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1887     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1888     {\tl_to_str:n a} {
1889       \bool_set_false:N \l_tmpa_bool
1890       \int_incr:N \l_tmpb_int
1891     }
1892     {\tl_to_str:n B} {
1893       \bool_set_false:N \l_tmpa_bool
1894       \int_incr:N \l_tmpb_int
1895     }
1896   }{
1897     \msg_set:nnn{stex}{error/wrongargs}{
1898       args~value~in~symbol~declaration~for~
1899       \l_stex_current_module_str ?
1900       \l_stex_symdecl_name_str ~
1901       needs~to~be~
1902       i,~a,~b~or~B,~but~##1~given
1903     }
1904     \msg_error:nn{stex}{error/wrongargs}
1905   }
1906 }
1907 \bool_if:NTF \l_tmpa_bool {
1908   % possibly numeric
1909   \str_if_empty:NTF \l_stex_symdecl_args_str {
1910     \prop_put:Nnn \l_tmpa_prop { args } {}
1911     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1912   }{
1913     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1914     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1915     \str_clear:N \l_tmpa_str
1916     \int_step_inline:nn \l_tmpa_int {
1917       \str_put_right:Nn \l_tmpa_str i
1918     }
1919     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1920   }
1921 } {
1922   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1923   \prop_put:Nnx \l_tmpa_prop { arity }
1924     { \str_count:N \l_stex_symdecl_args_str }
1925 }
1926 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1927
1928

```

```

1929 % semantic macro
1930
1931 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1932   \exp_args:Nx \stex_do_aftergroup:n {
1933     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1934       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1935     }}
1936   }
1937
1938   \bool_if:NF \l_stex_symdecl_local_bool {
1939     \exp_args:Nx \stex_add_to_current_module:n {
1940       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1941         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1942       } }
1943     }
1944   }
1945 }
1946
1947 % add to all symbols
1948
1949 \bool_if:NF \l_stex_symdecl_local_bool {
1950   \exp_args:Nx \stex_add_to_current_module:n {
1951     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1952       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1953     }
1954   }
1955   % \exp_args:Nx \stex_add_field_to_current_module:n {
1956   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1957   % }
1958 }
1959
1960 \stex_debug:nn{symbols}{New~symbol:~
1961   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1962   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1963   Args:~\prop_item:Nn \l_tmpa_prop { args }
1964 }
1965
1966 % circular dependencies require this:
1967
1968 \prop_if_exist:cF {
1969   l_stex_symdecl_
1970   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971   _prop
1972 } {
1973   \prop_set_eq:cN {
1974     l_stex_symdecl_
1975     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1976     _prop
1977   } \l_tmpa_prop
1978 }
1979
1980 \seq_clear:c {
1981   l_stex_symdecl_
1982   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

1983   _notations
1984 }
1985
1986 \bool_if:NF \l_stex_symdecl_local_bool {
1987   \exp_args:Nx
1988   \stex_add_to_current_module:n {
1989     \seq_clear:c {
1990       l_stex_symdecl_
1991       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1992       _notations
1993     }
1994     \prop_set_from_keyval:cn {
1995       l_stex_symdecl_
1996       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1997       _prop
1998     } {
1999       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2000       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2001       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2002       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2003       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2004       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2005     }
2006   }
2007 }
2008
2009 \stex_if_smsmode:TF {
2010   \bool_if:NF \l_stex_symdecl_local_bool {
2011     % \exp_args:Nx \stex_add_to_sms:n {
2012     %   \prop_set_from_keyval:cn {
2013     %     l_stex_symdecl_
2014     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2015     %     _prop
2016     %   } {
2017     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2018     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2019     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2020     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2021     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2022     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2023     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2024     %   }
2025     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2026     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027     %   }
2028     % }
2029   }
2030 }{
2031   \exp_args:Nx \stex_do_aftergroup:n {
2032     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2033       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2034     }
2035   }
2036   \stex_if_do_html:T {

```

```

2037 \stex_annotate_invisible:nnn {symdecl} {
2038   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2039 } {
2040   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2041   \stex_annotate_invisible:nnn{args}{}{
2042     \prop_item:Nn \l_tmpa_prop { args }
2043   }
2044   \stex_annotate_invisible:nnn{macroname}{#1}{}
2045   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2046     \stex_annotate_invisible:nnn{definiens}{}
2047     { $\l_stex_symdecl_definiens_tl$ }
2048   }
2049 }
2050 }
2051 }
2052 }

```

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

`\stex_get_symbol:n`

```

2053 \str_new:N \l_stex_get_symbol_uri_str
2054
2055 \cs_new_protected:Nn \stex_get_symbol:n {
2056   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2057     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2058   }{
2059     % argument is a string
2060     % is it a command name?
2061     \cs_if_exist:cTF { #1 }{
2062       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2063       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2064       \str_if_empty:NNTF \l_tmpa_str {
2065         \exp_args:Nx \cs_if_eq:NNTF {
2066           \tl_head:N \l_tmpa_tl
2067         } \stex_invoke_symbol:n {
2068           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2069         }{
2070           \__stex_symdecl_get_symbol_from_string:n { #1 }
2071         }
2072       } {
2073         \__stex_symdecl_get_symbol_from_string:n { #1 }
2074       }
2075     }{
2076       % argument is not a command name
2077       \__stex_symdecl_get_symbol_from_string:n { #1 }
2078       % \l_stex_all_symbols_seq
2079     }
2080   }
2081 }
2082
2083 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2084   \str_set:Nn \l_tmpa_str { #1 }
2085   \bool_set_false:N \l_tmpa_bool
2086   \stex_if_in_module:T {

```



```

2087 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2088 \bool_set_true:N \l_tmpa_bool
2089 \str_set:Nx \l_stex_get_symbol_uri_str {
2090 \l_stex_current_module_str ? #1
2091 }
2092 }
2093 }
2094 \bool_if:NF \l_tmpa_bool {
2095 \tl_set:Nn \l_tmpa_tl {
2096 \msg_set:nnn{stex}{error/unknownsymbol}{
2097 No~symbol~#1~found!
2098 }
2099 \msg_error:nn{stex}{error/unknownsymbol}
2100 }
2101 \str_set:Nn \l_tmpa_str { #1 }
2102 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2103 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2104 \str_set:Nn \l_tmpb_str { ##1 }
2105 \str_if_eq:eeT { \l_tmpa_str } {
2106 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2107 } {
2108 \seq_map_break:n {
2109 \tl_set:Nn \l_tmpa_tl {
2110 \str_set:Nn \l_stex_get_symbol_uri_str {
2111 ##1
2112 }
2113 }
2114 }
2115 }
2116 }
2117 \l_tmpa_tl
2118 }
2119 }
2120
2121 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2122 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2123 { \tl_tail:N \l_tmpa_tl }
2124 \tl_if_single:NTF \l_tmpa_tl {
2125 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2126 \exp_after:wN \str_set:Nn \exp_after:wN
2127 \l_stex_get_symbol_uri_str \l_tmpa_tl
2128 }{
2129 % TODO
2130 % tail is not a single group
2131 }
2132 }{
2133 % TODO
2134 % tail is not a single group
2135 }
2136 }

```

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

30.2 Notations

```

2137 <@@=stex_notation>

      notation arguments:
2138 \keys_define:nn { stex / notation } {
2139   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2140   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2141   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2142   op        .tl_set:N   = \l__stex_notation_op_tl ,
2143   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2144   primary   .default:n  = {true} ,
2145   unknown   .code:n     = \str_set:Nx
2146             \l__stex_notation_variant_str \l_keys_key_str
2147 }
2148
2149 \cs_new_protected:Nn \stex_notation_args:n {
2150   \str_clear:N \l__stex_notation_lang_str
2151   \str_clear:N \l__stex_notation_variant_str
2152   \str_clear:N \l__stex_notation_prec_str
2153   \tl_clear:N \l__stex_notation_op_tl
2154   \bool_set_false:N \l__stex_notation_primary_bool
2155
2156   \keys_set:nn { stex / notation } { #1 }
2157 }

```

\notation

```

2158 \NewDocumentCommand \notation { 0{ } m } {
2159   \stex_notation_args:n { #1 }
2160   \tl_clear:N \l_stex_symdecl_definiens_tl
2161   \stex_get_symbol:n { #2 }
2162   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2163 }
2164 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2165 \cs_new_protected:Nn \stex_notation_do:nn {
2166   \let\l_stex_current_symbol_str\relax
2167   \prop_set_eq:Nc \l_tmpa_prop {
2168     l_stex_symdecl_ #1 _prop
2169   }
2170
2171   \prop_clear:N \l_tmpb_prop
2172   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2173   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2174   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2175
2176   % precedences
2177   \seq_clear:N \l_tmpb_seq
2178   \exp_args:NNno
2179   \str_if_empty:NTF \l__stex_notation_prec_str {
2180     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2181     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2182     \exp_args:NNx
2183     \prop_put:Nno \l_tmpb_prop { opprec }
2184     { \neginfprec }
2185   }{
2186     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2187   }
2188 } {
2189   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2190     \exp_args:NNx
2191     \prop_put:Nno \l_tmpb_prop { opprec }
2192     { \neginfprec }
2193     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2194     \int_step_inline:nn { \l_tmpa_str } {
2195       \exp_args:NNx
2196       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2197     }
2198   }{
2199     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2200     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2201       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2202       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2203         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2204         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2205         \seq_map_inline:Nn \l_tmpa_seq {
2206           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2207         }
2208       }
2209       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2210     }{
2211       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2212       \int_compare:nNnTF \l_tmpa_str = 0 {
2213         \exp_args:NNx
2214         \prop_put:Nno \l_tmpb_prop { opprec }
2215         { \infprec }
2216       }{
2217         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2218       }
2219     }
2220   }
2221 }
2222
2223 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2224 \int_step_inline:nn { \l_tmpa_str } {
2225   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2226     \exp_args:NNx
2227     \seq_put_right:Nn \l_tmpb_seq {
2228       \prop_item:Nn \l_tmpb_prop { opprec }
2229     }
2230   }
2231 }
2232
2233 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2234 \tl_clear:N \l_tmpa_tl
2235

```

```

2236 \int_compare:nNnTF \l_tmpa_str = 0 {
2237   \exp_args:NNe
2238   \cs_set:Npn \l__stex_notation_macrocode_cs {
2239     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2240     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2241     { \prop_item:Nn \l_tmpb_prop { opprec } }
2242     { \exp_not:n { #2 } }
2243   }
2244   \__stex_notation_final:
2245 }{
2246   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2247   \str_if_in:NnTF \l_tmpb_str b {
2248     \exp_args:Nne \use:nn
2249     {
2250       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2251       \cs_set:Npn \l_tmpa_str } { {
2252         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2253         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2254         { \prop_item:Nn \l_tmpb_prop { opprec } }
2255         { \exp_not:n { #2 } }
2256       }}
2257   }{
2258     \str_if_in:NnTF \l_tmpb_str B {
2259       \exp_args:Nne \use:nn
2260       {
2261         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2262         \cs_set:Npn \l_tmpa_str } { {
2263           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2264           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2265           { \prop_item:Nn \l_tmpb_prop { opprec } }
2266           { \exp_not:n { #2 } }
2267         } }
2268     }{
2269       \exp_args:Nne \use:nn
2270       {
2271         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2272         \cs_set:Npn \l_tmpa_str } { {
2273           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2274           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2275           { \prop_item:Nn \l_tmpb_prop { opprec } }
2276           { \exp_not:n { #2 } }
2277         } }
2278     }
2279   }
2280
2281   \int_zero:N \l_tmpa_int
2282   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2283   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2284   \__stex_notation_arguments:
2285 }
2286 }

```

(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

```

2287 \cs_new_protected:Nn \_stex_notation_arguments: {
2288   \int_incr:N \l_tmpa_int
2289   \str_if_empty:NTF \l_tmpa_str {
2290     \_stex_notation_final:
2291   }{
2292     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2293     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2294     \str_if_eq:VnTF \l_tmpb_str a {
2295       \_stex_notation_argument_assoc:n
2296     }{
2297       \str_if_eq:VnTF \l_tmpb_str B {
2298         \_stex_notation_argument_assoc:n
2299       }{
2300         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2301         \tl_put_right:Nx \l_tmpa_tl {
2302           { \_stex_term_math_arg:nnn
2303             { \int_use:N \l_tmpa_int }
2304             { \l_tmpb_str }
2305             { ####\int_use:N \l_tmpa_int }
2306           }
2307         }
2308         \_stex_notation_arguments:
2309       }
2310     }
2311   }
2312 }
```

(End definition for `_stex_notation_arguments:.`)

`_stex_notation_argument_assoc:n`

```

2313 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2314   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2315   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2316   \tl_put_right:Nx \l_tmpa_tl {
2317     { \_stex_term_math_assoc_arg:nnnn
2318       { \int_use:N \l_tmpa_int }
2319       { \l_tmpb_str }
2320       \exp_args:No \exp_not:n
2321       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2322       { ####\int_use:N \l_tmpa_int }
2323     }
2324   }
2325   \_stex_notation_arguments:
2326 }
```

(End definition for `_stex_notation_argument_assoc:n.`)

`_stex_notation_final:` Called after processing all notation arguments

```

2327 \cs_new_protected:Nn \_stex_notation_final: {
2328   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2329   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2330   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2331   \exp_args:Nne \use:nn
```

```

2332 {
2333 \cs_generate_from_arg_count:cNnn {
2334   stex_notation_ \l_tmpa_str \c_hash_str
2335   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2336   _cs
2337 }
2338 \cs_set:Npn \l_tmpb_str } { {
2339   \exp_after:wN \exp_after:wN \exp_after:wN
2340   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2341   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2342 } }
2343
2344 \tl_if_empty:NF \l__stex_notation_op_tl {
2345   \cs_set:cpx {
2346     stex_op_notation_ \l_tmpa_str \c_hash_str
2347     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2348     _cs
2349   } {
2350     \_stex_term_oms:nnn {
2351       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2352       \l__stex_notation_lang_str
2353     }{
2354       \l_tmpa_str
2355     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2356   }
2357 }
2358
2359 \exp_args:Ne
2360 \stex_add_to_current_module:n {
2361   \cs_generate_from_arg_count:cNnn {
2362     stex_notation_ \l_tmpa_str \c_hash_str
2363     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2364     _cs
2365   } \cs_set:Npn {\l_tmpb_str} {
2366     \exp_after:wN \exp_after:wN \exp_after:wN
2367     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2368     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2369   }
2370   \tl_if_empty:NF \l__stex_notation_op_tl {
2371     \cs_set:cpn {
2372       stex_op_notation_ \l_tmpa_str \c_hash_str
2373       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2374       _cs
2375     } {
2376       \_stex_term_oms:nnn {
2377         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2378         \l__stex_notation_lang_str
2379       }{
2380         \l_tmpa_str
2381       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2382     }
2383   }
2384 }
2385

```

```

2386 \seq_put_right:cx {
2387   l_stex_symdecl_
2388   \prop_item:Nn \l_tmpb_prop { symbol }
2389   _notations
2390 } {
2391   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2392 }
2393
2394 \stex_debug:nn{symbols}{
2395   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2396   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2397   Operator~precedence:~
2398   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2399   Argument~precedences:~
2400   \seq_use:Nn \l_tmpa_seq {,~}^^J
2401   Notation: \cs_meaning:c {
2402     stex_notation_ \l_tmpa_str \c_hash_str
2403     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2404     _cs
2405   }
2406 }
2407
2408 \prop_set_eq:cN {
2409   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2410   \c_hash_str \l__stex_notation_lang_str _prop
2411 } \l_tmpb_prop
2412
2413 \exp_args:Ne
2414 \stex_add_to_current_module:n {
2415   \seq_put_right:cn {
2416     l_stex_symdecl_
2417     \prop_item:Nn \l_tmpb_prop { symbol }
2418     _notations
2419   } {
2420     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2421   }
2422   \prop_set_from_keyval:cn {
2423     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2424     \c_hash_str \l__stex_notation_lang_str _prop
2425   } {
2426     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2427     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2428     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2429     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2430     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2431   }
2432 }
2433
2434 \stex_if_smsmode:TF {
2435   \stex_smsmode_set_codes:
2436   % \exp_args:Nx \stex_add_to_sms:n {
2437   %   \prop_set_from_keyval:cn {
2438   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2439   %     \c_hash_str \l__stex_notation_lang_str _prop

```

```

2440 %      } {
2441 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2442 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2443 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2444 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2445 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2446 %      }
2447 %  }
2448 }{
2449
2450 % HTML annotations
2451 \stex_if_do_html:T {
2452     \stex_annotate_invisible:nnn { notation }
2453     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2454         \stex_annotate_invisible:nnn { notationfragment }
2455         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2456         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2457         \stex_annotate_invisible:nnn { precedence }
2458         { \prop_item:Nn \l_tmpb_prop { opprec };
2459           \seq_use:Nn \l_tmpa_seq { x }
2460         }{}
2461
2462         \int_zero:N \l_tmpa_int
2463         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2464         \tl_clear:N \l_tmpa_tl
2465         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2466             \int_incr:N \l_tmpa_int
2467             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2468             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2469             \str_if_eq:VnTF \l_tmpb_str a {
2470                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2471                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2472                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2473                 } }
2474             }{
2475                 \str_if_eq:VnTF \l_tmpb_str B {
2476                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2477                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2478                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2479                     } }
2480                 }{
2481                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2482                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2483                     } }
2484                 }
2485             }
2486         }
2487         \stex_annotate_invisible:nnn { notationcomp }{}{
2488             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2489             $ \exp_args:Nno \use:nn { \use:c {
2490                 stex_notation_ \l_stex_current_symbol_str
2491                 \c_hash_str \l__stex_notation_variant_str
2492                 \c_hash_str \l__stex_notation_lang_str _cs
2493             } } { \l_tmpa_tl } $

```



```

2494     }
2495   }
2496 }
2497 }
2498 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

2499 \keys_define:nn { stex / setnotation } {
2500   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2501   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2502   unknown .code:n = \str_set:Nx
2503     \l__stex_notation_variant_str \l_keys_key_str
2504 }
2505
2506 \cs_new_protected:Nn \_stex_setnotation_args:n {
2507   \str_clear:N \l__stex_notation_lang_str
2508   \str_clear:N \l__stex_notation_variant_str
2509   \keys_set:nn { stex / setnotation } { #1 }
2510 }
2511
2512 \NewDocumentCommand \setnotation {m m} {
2513   \stex_get_symbol:n { #1 }
2514   \_stex_setnotation_args:n { #2 }
2515   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2516     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2517     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2518       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2519     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2520       { \c_hash_str }
2521     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2522       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2523     \exp_args:Nx \stex_add_to_current_module:n {
2524       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2525         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2526     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2527       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2528     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2529       { \c_hash_str }
2530   }
2531   \stex_debug:nn {notations}{
2532     Setting~default~notation~
2533     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2534     \l_stex_get_symbol_uri_str \
2535     \expandafter\meaning\csname
2536     l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2537   }
2538   }{
2539     % todo throw error
2540   }
2541 }
2542

```

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

\symdef

```
2543 \keys_define:nn { stex / symdef } {
2544   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2545   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2546   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2547   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2548   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2549   op        .tl_set:N    = \l__stex_notation_op_tl ,
2550   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2551   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2552   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2553   unknown   .code:n      = \str_set:Nx
2554             \l__stex_notation_variant_str \l_keys_key_str
2555 }
2556
2557 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2558   \str_clear:N \l_stex_symdecl_name_str
2559   \str_clear:N \l_stex_symdecl_args_str
2560   \bool_set_false:N \l_stex_symdecl_local_bool
2561   \tl_clear:N \l_stex_symdecl_type_tl
2562   \tl_clear:N \l_stex_symdecl_definiens_tl
2563   \str_clear:N \l__stex_notation_lang_str
2564   \str_clear:N \l__stex_notation_variant_str
2565   \str_clear:N \l__stex_notation_prec_str
2566   \tl_clear:N \l__stex_notation_op_tl
2567
2568   \keys_set:nn { stex / symdef } { #1 }
2569 }
2570
2571 \NewDocumentCommand \symdef { 0{} m } {
2572   \__stex_notation_symdef_args:n { #1 }
2573   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2574   \stex_symdecl_do:n { #2 }
2575   \exp_args:Nx \stex_notation_do:nn {
2576     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2577   }
2578 }
2579 \stex_deactivate_macro:Nn \symdef {module~environments}
2580
2581 (End definition for \symdef. This function is documented on page 37.)
2582 \endpackage
```

Chapter 31

STEX -Terms Implementation

```
2581 <*package>
2582
2583 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2584
2585 <@@=stex_terms>
2586
2587 Warnings and error messages
2588 \msg_new:nnn{stex}{error/nonotation}{
2589   Symbol~#1~invoked,~but~has~no~notation~#2!
2590 }
2591 \msg_new:nnn{stex}{error/notationarg}{
2592   Error~in~parsing~notation~#1
2593 }
2594 \msg_new:nnn{stex}{error/noop}{
2595   Symbol~#1~has~no~operator~notation~for~notation~#2
2596 }
```

31.1 Symbol Invocations

Arguments:

```
2596 \keys_define:nn { stex / terms } {
2597   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2598   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2599   unknown .code:n = \str_set:Nx
2600     \l__stex_terms_variant_str \l_keys_key_str
2601 }
2602
2603 \cs_new_protected:Nn \__stex_terms_args:n {
2604   \str_clear:N \l__stex_terms_lang_str
2605   \str_clear:N \l__stex_terms_variant_str
2606   \str_clear:N \l__stex_terms_prec_str
2607   \tl_clear:N \l__stex_terms_op_tl
2608 }
2609 \keys_set:nn { stex / terms } { #1 }
```

2610 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2611 \cs_new_protected:Nn \stex_invoke_symbol:n {
2612   \if_mode_math:
2613     \exp_after:wN \__stex_terms_invoke_math:n
2614   \else:
2615     \exp_after:wN \__stex_terms_invoke_text:n
2616   \fi: { #1 }
2617 }
```

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

__stex_terms_invoke_math:n

```
2618 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2619   \peek_charcode_remove:NTF ! {
2620     \peek_charcode:NTF [ {
2621       \__stex_terms_invoke_op:nw { #1 }
2622     }{
2623       \peek_charcode_remove:NTF ! {
2624         \peek_charcode:NTF [ {
2625           \__stex_terms_invoke_op_custom:nw
2626         }{
2627           % TODO throw error
2628         }
2629       }{
2630         \__stex_terms_invoke_op:nw { #1 } []
2631       }
2632     }
2633   }{
2634     \peek_charcode_remove:NTF * {
2635       \__stex_terms_invoke_text:n { #1 }
2636     }{
2637       \peek_charcode:NTF [ {
2638         \__stex_terms_invoke_math:nw { #1 }
2639       }{
2640         \__stex_terms_invoke_math:nw { #1 } []
2641       }
2642     }
2643   }
2644 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2645 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2646   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2647     \stex_highlight_term:nn{#1}{#2}
2648   }
2649 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

__stex_terms_invoke_op:nw

```

2650 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2651   \__stex_terms_args:n { #2 }
2652   \cs_if_exist:cTF {
2653     stex_op_notation_ #1 \c_hash_str
2654     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2655   }{
2656     \csname stex_op_notation_ #1 \c_hash_str
2657       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2658     \endcsname
2659   }{
2660     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2661   }
2662 }

```

(End definition for __stex_terms_invoke_op:nw.)

__stex_terms_invoke_math:nw

```

2663 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2664   \__stex_terms_args:n { #2 }
2665   \seq_if_empty:cTF {
2666     l_stex_symdecl_ #1 _notations
2667   } {
2668     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2669   } {
2670     \seq_if_in:cxTF {
2671       l_stex_symdecl_ #1 _notations
2672     }
2673     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2674       \str_set:Nn \l_stex_current_symbol_str { #1 }
2675       \use:c{
2676         stex_notation_ #1 \c_hash_str
2677         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2678         _cs
2679       }
2680     }{
2681       \str_if_empty:NTF \l__stex_terms_variant_str {
2682         \str_if_empty:NTF \l__stex_terms_lang_str {
2683           \seq_get_left:cN {
2684             l_stex_symdecl_ #1 _notations
2685           } \l_tmpa_str
2686           \str_set:Nn \l_stex_current_symbol_str { #1 }
2687           \use:c{
2688             stex_notation_ #1 \c_hash_str \l_tmpa_str
2689             _cs
2690           }
2691         }{
2692           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2693             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2694           }
2695         }
2696       }{
2697         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2698           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2699     }
2700   }
2701 }
2702 }
2703 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2704 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2705   \peek_charcode_remove:NTF ! {
2706     \stex_term_custom:nn { #1 } { }
2707   }{
2708     \prop_set_eq:Nc \l_tmpa_prop {
2709       l_stex_symdecl_ #1 _prop
2710     }
2711     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2712     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2713   }
2714 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2715 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2716 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2717 \int_new:N \l__stex_terms_downprec
2718 \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
2719 \tl_set:Nn \l__stex_terms_left_bracket_str (
2720 \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

```

2721 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2722   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2723     \bool_set_false:N \l__stex_terms_brackets_done_bool
2724     #2
2725   } {
2726     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2727       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2728         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2729         \dobrackets { #2 }
2730       }

```

```

2731     }{ #2 }
2732   }
2733 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2734 \bool_new:N \l__stex_terms_brackets_done_bool
2735 %\RequirePackage{scalerel}
2736 \cs_new_protected:Npn \dobrackets #1 {
2737   %\ThisStyle{\if D\m@switch
2738   %   \exp_args:Nnx \use:nn
2739   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2740   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2741   % \else
2742   \exp_args:Nnx \use:nn
2743   {
2744     \bool_set_true:N \l__stex_terms_brackets_done_bool
2745     \int_set:Nn \l__stex_terms_downprec \infprec
2746     \l__stex_terms_left_bracket_str
2747     #1
2748   }
2749   {
2750     \bool_set_false:N \l__stex_terms_brackets_done_bool
2751     \l__stex_terms_right_bracket_str
2752     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2753   }
2754   %\fi}
2755 }

```

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

`\withbrackets`

```

2756 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2757   \exp_args:Nnx \use:nn
2758   {
2759     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2760     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2761     #3
2762   }
2763   {
2764     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2765     {\l__stex_terms_left_bracket_str}
2766     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2767     {\l__stex_terms_right_bracket_str}
2768   }
2769 }

```

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

`\STEXinvisible`

```

2770 \cs_new_protected:Npn \STEXinvisible #1 {
2771   \stex_annotate_invisible:n { #1 }
2772 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2773 \cs_new_protected:Nn \_stex_term_oms:nnn {
2774   \stex_annotate:nnn{ OMID }{ #2 }{
2775     \stex_highlight_term:nn { #1 } { #3 }
2776   }
2777 }
2778
2779 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2780   \__stex_terms_maybe_brackets:nn { #3 }{
2781     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2782   }
2783 }

```

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

`_stex_term_math_oma:nnnn`

```

2784 \cs_new_protected:Nn \_stex_term_oma:nnn {
2785   \stex_annotate:nnn{ OMA }{ #2 }{
2786     \stex_highlight_term:nn { #1 } { #3 }
2787   }
2788 }
2789
2790 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2791   \__stex_terms_maybe_brackets:nn { #3 }{
2792     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2793   }
2794 }

```

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

`_stex_term_math_omb:nnnn`

```

2795 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2796   \stex_annotate:nnn{ OMBIND }{ #2 }{
2797     \stex_highlight_term:nn { #1 } { #3 }
2798   }
2799 }
2800
2801 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2802   \__stex_terms_maybe_brackets:nn { #3 }{
2803     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2804   }
2805 }

```

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

`_stex_term_math_arg:nnn`

```

2806 \cs_new_protected:Nn \_stex_term_arg:nn {
2807   \stex_unhighlight_term:n {
2808     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2809   }
2810 }

```



```

2811 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2812   \exp_args:Nnx \use:nn
2813     { \int_set:Nn \l__stex_terms_downprec { #2 }
2814       \stex_term_arg:nn { #1 }{ #3 }
2815     }
2816     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2817   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2818 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2819   \clist_set:Nn \l_tmpa_clist{ #4 }
2820   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2821     \tl_set:Nn \l_tmpa_tl { #4 }
2822   }{
2823     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2824     \clist_reverse:N \l_tmpa_clist
2825     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2826
2827     \clist_map_inline:Nn \l_tmpa_clist {
2828       \exp_args:NNNo \exp_args:Nno \tl_set:Nn \l_tmpa_tl {
2829         \exp_args:Nno
2830           \l_tmpa_cs { ##1 } \l_tmpa_tl
2831       }
2832     }
2833
2834   }
2835   \exp_args:Nnno
2836   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2837 }

```

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

`\stex_term_custom:nn`

```

2838 \cs_new_protected:Nn \stex_term_custom:nn {
2839   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2840   \str_set:Nn \l_tmpa_str { #2 }
2841   \tl_clear:N \l_tmpa_tl
2842   \int_zero:N \l_tmpa_int
2843   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2844   \__stex_terms_custom_loop:
2845 }

```

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

`__stex_terms_custom_loop:`

```

2846 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2847   \bool_set_false:N \l_tmpa_bool
2848   \bool_while_do:nn {
2849     \str_if_eq_p:ee X {
2850       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2851     }
2852   }{
2853     \int_incr:N \l_tmpa_int

```

```

2854 }
2855
2856 \peek_charcode:NTF [ {
2857   % notation/text component
2858   \__stex_terms_custom_component:w
2859 } {
2860   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2861     % all arguments read => finish
2862     \__stex_terms_custom_final:
2863   } {
2864     % arguments missing
2865     \peek_charcode_remove:NTF * {
2866       % invisible, specific argument position or both
2867       \peek_charcode:NTF [ {
2868         % visible specific argument position
2869         \__stex_terms_custom_arg:wn
2870       } {
2871         % invisible
2872         \peek_charcode_remove:NTF * {
2873           % invisible specific argument position
2874           \__stex_terms_custom_arg_inv:wn
2875         } {
2876           % invisible next argument
2877           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2878         }
2879       }
2880     } {
2881       % next normal argument
2882       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2883     }
2884   }
2885 }
2886 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2887 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2888   \bool_set_true:N \l_tmpa_bool
2889   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2890 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2891 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2892   \str_set:Nx \l_tmpb_str {
2893     \str_item:Nn \l_tmpa_str { #1 }
2894   }
2895   \str_case:VnTF \l_tmpb_str {
2896     { X } {
2897       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2898     }
2899     { i } { \__stex_terms_custom_set_X:n { #1 } }
2900     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2901     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2902     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2903   }{}{
2904     \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2905   }
2906
2907   \bool_if:nTF \l_tmpa_bool {
2908     \tl_put_right:Nx \l_tmpa_tl {
2909       \stex_annotate_invisible:n {
2910         \stex_term_arg:nn { \int_eval:n { #1 } }
2911         \exp_not:n { { #2 } }
2912       }
2913     }
2914   } {
2915     \tl_put_right:Nx \l_tmpa_tl {
2916       \stex_term_arg:nn { \int_eval:n { #1 } }
2917       \exp_not:n { { #2 } }
2918     }
2919   }
2920
2921   \_stex_terms_custom_loop:
2922 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2923 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2924   \str_set:Nx \l_tmpa_str {
2925     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2926     X
2927     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2928   }
2929 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2930 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2931   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2932   \_stex_terms_custom_loop:
2933 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2934 \cs_new_protected:Nn \_stex_terms_custom_final: {
2935   \int_compare:nNnTF \l_tmpb_int = 0 {
2936     \exp_args:Nnno \stex_term_oms:nnn
2937   }{
2938     \str_if_in:NnTF \l_tmpa_str {b} {
2939       \exp_args:Nnno \stex_term_ombind:nnn
2940     } {
2941       \exp_args:Nnno \stex_term_oma:nnn
2942     }
2943   }

```

```

2944 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2945 }

```

(End definition for `_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2946 \NewDocumentCommand \symref { m m }{
2947   \let\compemph_uri_prev:\compemph@uri
2948   \let\compemph@uri\symrefemph@uri
2949   \STEXsymbol{#1}! [#2]
2950   \let\compemph@uri\compemph_uri_prev:
2951 }
2952
2953 \keys_define:nn { stex / symname } {
2954   post      .str_set_x:N    = \l_stex_symname_post_str
2955 }
2956
2957 \cs_new_protected:Nn \stex_symname_args:n {
2958   \str_clear:N \l_stex_symname_post_str
2959   \keys_set:nn { stex / symname } { #1 }
2960 }
2961
2962 \NewDocumentCommand \symname { 0{} m }{
2963   \stex_symname_args:n { #1 }
2964   \stex_get_symbol:n { #2 }
2965   \str_set:Nx \l_tmpa_str {
2966     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2967   }
2968   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2969
2970   \let\compemph_uri_prev:\compemph@uri
2971   \let\compemph@uri\symrefemph@uri
2972   \exp_args:NNx \use:nn
2973   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2974     \l_tmpa_str \l_stex_symname_post_str
2975   ] }
2976   \let\compemph@uri\compemph_uri_prev:
2977 }

```

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

31.3 Notation Components

```

2978 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2979
2980 \str_new:N \l_stex_current_symbol_str
2981 \cs_new_protected:Nn \stex_highlight_term:nn {
2982   \exp_args:Nnx
2983   \use:nn {
2984     \str_set:Nx \l_stex_current_symbol_str { #1 }
2985     #2
2986   } {

```

```

2987 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2988 { \l_stex_current_symbol_str }
2989 }
2990 }
2991
2992 \cs_new_protected:Nn \stex_unhighlight_term:n {
2993 % \latexml_if:TF {
2994 % #1
2995 % } {
2996 % \rustex_if:TF {
2997 % #1
2998 % } {
2999 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3000 % }
3001 % }
3002 }

```

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

```

\comp
\compemph@uri 3003 \cs_new_protected:Npn \comp #1 {
\compemph 3004 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3005 \rustex_if:TF {
\defemph@uri 3006 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3007 }{
\symrefemph@uri 3008 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3009 }
3010 }
3011 }
3012
3013 \cs_new_protected:Npn \compemph@uri #1 #2 {
3014 \compemph{ #1 }
3015 }
3016
3017
3018 \cs_new_protected:Npn \compemph #1 {
3019 #1
3020 }
3021
3022 \cs_new_protected:Npn \defemph@uri #1 #2 {
3023 \defemph{#1}
3024 }
3025
3026 \cs_new_protected:Npn \defemph #1 {
3027 \textbf{#1}
3028 }
3029
3030 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3031 \symrefemph{#1}
3032 }
3033
3034 \cs_new_protected:Npn \symrefemph #1 {
3035 \textbf{#1}
3036 }

```

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

`\ellipses`

```
3037 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 3038 \bool_new:N \l_stex_inarray_bool
\parrayline 3039 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3040 \NewDocumentCommand \parray { m m } {
\parraycell 3041 \begin{group}
3042 \bool_set_true:N \l_stex_inarray_bool
3043 \begin{array}{#1}
3044 #2
3045 \end{array}
3046 \end{group}
3047 }
3048
3049 \NewDocumentCommand \prmatrix { m } {
3050 \begin{group}
3051 \bool_set_true:N \l_stex_inarray_bool
3052 \begin{matrix}
3053 #1
3054 \end{matrix}
3055 \end{group}
3056 }
3057
3058 \def \maybepline {
3059 \bool_if:NT \l_stex_inarray_bool {\hline}
3060 }
3061
3062 \def \parrayline #1 #2 {
3063 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3064 }
3065
3066 \def \pmrow #1 { \parrayline{}{ #1 } }
3067
3068 \def \parraylineh #1 #2 {
3069 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3070 }
3071
3072 \def \parraycell #1 {
3073 #1 \bool_if:NT \l_stex_inarray_bool {&}
3074 }

```

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

```
3075 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
3076 <*package>
3077
3078 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3079
3080 <@@=stex_features>
3081
3082   Warnings and error messages
3083 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3084   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3085 }
3086 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3087   Symbol~#1~not~assigned~in~interpretmodule~#2
3088 }
3089
```

32.1 Imports with modification

```
3088 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3089   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3090     \__stex_features_get_symbol_from_cs:n { #1 }
3091   }{
3092     % argument is a string
3093     % is it a command name?
3094     \cs_if_exist:cTF { #1 }{
3095       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3096       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3097       \str_if_empty:NNTF \l_tmpa_str {
3098         \exp_args:Nx \cs_if_eq:NNTF {
3099           \tl_head:N \l_tmpa_tl
3100         } \stex_invoke_symbol:n {
3101           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3102         }{
3103           \__stex_features_get_symbol_from_string:n { #1 }
3104         }
3105       }
3106     }
3107   }
3108 }
```

```

3104     }
3105   } {
3106     \__stex_features_get_symbol_from_string:n { #1 }
3107   }
3108   ){
3109     % argument is not a command name
3110     \__stex_features_get_symbol_from_string:n { #1 }
3111     % \l_stex_all_symbols_seq
3112   }
3113 }
3114 }
3115
3116 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3117   \str_set:Nn \l_tmpa_str { #1 }
3118   \bool_set_false:N \l_tmpa_bool
3119   \bool_if:NF \l_tmpa_bool {
3120     \tl_set:Nn \l_tmpa_tl {
3121       \msg_set:nnn{stex}{error/unknownsymbol}{
3122         No~symbol~#1~found!
3123       }
3124       \msg_error:nn{stex}{error/unknownsymbol}
3125     }
3126     \str_set:Nn \l_tmpa_str { #1 }
3127     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3128     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3129       \str_set:Nn \l_tmpb_str { ##1 }
3130       \str_if_eq:eeT { \l_tmpa_str } {
3131         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3132       } {
3133         \seq_map_break:n {
3134           \tl_set:Nn \l_tmpa_tl {
3135             \str_set:Nn \l_stex_get_symbol_uri_str {
3136               ##1
3137             }
3138             \__stex_features_get_symbol_check:
3139           }
3140         }
3141       }
3142     }
3143     \l_tmpa_tl
3144   }
3145 }
3146
3147 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3148   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3149   { \tl_tail:N \l_tmpa_tl }
3150   \tl_if_single:NTF \l_tmpa_tl {
3151     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3152       \exp_after:wN \str_set:Nn \exp_after:wN
3153       \l_stex_get_symbol_uri_str \l_tmpa_tl
3154       \__stex_features_get_symbol_check:
3155     }{
3156       % TODO
3157       % tail is not a single group

```



```

3158     }
3159 }{
3160     % TODO
3161     % tail is not a single group
3162 }
3163 }
3164
3165 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3166     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3167     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3168         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3169         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3170         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3171             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3172                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3173             }
3174         }
3175     }{
3176         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3177             \l_stex_current_copymodule_name_str~(inexplicitly)
3178         }
3179     }
3180 }
3181
3182 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3183     \stex_import_module_uri:nn { #1 } { #2 }
3184     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3185     \stex_import_require_module:nnnn
3186     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3187     { \l_stex_import_path_str } { \l_stex_import_name_str }
3188     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3189     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3190     \seq_clear:N \l__stex_features_copymodule_fields_seq
3191     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3192         \seq_map_inline:cn {c_stex_module_###1_constants}{
3193             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3194                 ###1 ? #####1
3195             }
3196         }
3197     }
3198     \seq_clear:N \l_tmpa_seq
3199     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3200         name      = \l_stex_current_copymodule_name_str ,
3201         module    = \l_stex_current_module_str ,
3202         from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3203         includes  = \l_tmpa_seq ,
3204         fields    = \l_tmpa_seq
3205     }
3206     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3207         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3208     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3209     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3210     \stex_if_smsmode:TF {
3211         \stex_smsmode_set_codes:

```

```

3212 } {
3213   \begin{stex_annotate_env} {#4} {
3214     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3215   }
3216   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3217 }
3218 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3219 \bool_set_false:N \l_stex_html_do_output_bool
3220 }
3221 \cs_new_protected:Nn \stex_copymodule_end:n {
3222   \def \l_tmpa_cs ##1 ##2 {#1}
3223   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3224   \tl_clear:N \l_tmpa_tl
3225   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3226   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3227     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3228       \l_tmpa_cs{##1}{####1}
3229       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3230         \tl_put_right:Nx \l_tmpa_tl {
3231           \prop_set_from_keyval:cn {
3232             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3233           }{
3234             \exp_after:wN \prop_to_keyval:N \csname
3235               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3236             \endcsname
3237           }
3238           \seq_clear:c {
3239             l_stex_symdecl_
3240             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3241             _notations
3242           }
3243         }
3244         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3245         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3246         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3247           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3248           \tl_put_right:Nx \l_tmpa_tl {
3249             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3250               \stex_invoke_symbol:n {
3251                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3252               }
3253             }
3254           }
3255         }
3256       }{
3257         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3258         \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3259         \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3260         \tl_put_right:Nx \l_tmpa_tl {
3261           \prop_set_from_keyval:cn {
3262             l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3263           }{
3264             \prop_to_keyval:N \l_tmpa_prop
3265           }

```

```

3266         \seq_clear:c {
3267             l_stex_symdecl_
3268             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3269             _notations
3270         }
3271     }
3272     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3273     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3274         \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3275         \tl_put_right:Nx \l_tmpa_tl {
3276             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3277                 \stex_invoke_symbol:n {
3278                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3279                 }
3280             }
3281         }
3282     }
3283 }
3284 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3285     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3286 }
3287 % todo notations
3288 }}
3289 }
3290 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3291 \tl_put_left:Nx \l_tmpa_tl {
3292     \prop_set_from_keyval:cn {
3293         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3294     }{
3295         \prop_to_keyval:N \l_stex_current_copymodule_prop
3296     }
3297 }
3298 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3299 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3300 \exp_args:Nx \stex_do_aftergroup:n {
3301     \exp_args:No \exp_not:n \l_tmpa_tl
3302 }
3303 \stex_if_smsmode:F {
3304     \end{stex_annotate_env}
3305 }
3306 }
3307
3308 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3309     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3310     \stex_deactivate_macro:Nn \symdecl {module~environments}
3311     \stex_deactivate_macro:Nn \symdef {module~environments}
3312     \stex_deactivate_macro:Nn \notation {module~environments}
3313     \stex_reactivate_macro:N \assign
3314     \stex_reactivate_macro:N \renamedec1
3315     \stex_reactivate_macro:N \donotcopy
3316 }{
3317     \stex_copymodule_end:n {}
3318 }
3319

```

```

3320 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3321   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3322   \stex_deactivate_macro:Nn \symdecl {module~environments}
3323   \stex_deactivate_macro:Nn \symdef {module~environments}
3324   \stex_deactivate_macro:Nn \notation {module~environments}
3325   \stex_reactivate_macro:N \assign
3326   \stex_reactivate_macro:N \renamedec1
3327   \stex_reactivate_macro:N \donotcopy
3328 }{
3329   \stex_copymodule_end:n {
3330     \tl_if_exist:cF {
3331       l__stex_features_copymodule_##1?##2_def_tl
3332     }{
3333       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3334         ##1?##2
3335       }{\l_stex_current_copymodule_name_str}
3336     }
3337   }
3338 }
3339
3340 \NewDocumentCommand \donotcopy { 0{} m}{
3341   \stex_import_module_uri:nn { #1 } { #2 }
3342   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3343   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3344     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3345     \seq_map_inline:cn {c_stex_module_##1_constants}{
3346       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? #####1 }
3347       \bool_lazy_any_p:nT {
3348         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3349         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3350         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3351       }{
3352         % TODO throw error
3353       }
3354     }
3355   }
3356
3357   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3358   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3359   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3360 }
3361
3362 \NewDocumentCommand \assign { m m }{
3363   \stex_get_symbol_in_copymodule:n {#1}
3364   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3365   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3366 }
3367
3368 \keys_define:nn { stex / renamedec1 } {
3369   name .str_set_x:N = \l_stex_renamedec1_name_str
3370 }
3371 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3372   \str_clear:N \l_stex_renamedec1_name_str
3373

```

```

3374 \keys_set:nn { stex / renamedec1 } { #1 }
3375 }
3376
3377 \NewDocumentCommand \renamedec1 { 0{} m m}{
3378   \__stex_features_renamedec1_args:n { #1 }
3379   \stex_get_symbol_in_copymodule:n {#2}
3380   \stex_debug:nn{renamedec1}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3381   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3382   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3383     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3384       \l_stex_get_symbol_uri_str
3385     } }
3386   } {
3387     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3388     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3389     \prop_set_eq:cc {l_stex_symdecl_
3390       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3391     _prop
3392     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3393     \seq_set_eq:cc {l_stex_symdecl_
3394       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3395     _notations
3396     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3397     \prop_put:cnx {l_stex_symdecl_
3398       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3399     _prop
3400     }{ name }{ \l_stex_renamedec1_name_str }
3401     \prop_put:cnx {l_stex_symdecl_
3402       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3403     _prop
3404     }{ module }{ \l_stex_current_module_str }
3405     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3406       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3407     }
3408     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3409       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3410     } }
3411   }
3412 }
3413 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3414 %   \_stex_notation_args:n { #1 }
3415 %   \tl_clear:N \l_stex_symdecl_definiens_tl
3416 %   \stex_get_symbol_in_copymodule:n { #2 }
3417 %   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3418 %   % todo
3419 %}
3420 \stex_deactivate_macro:Nn \assign {copymodules}
3421 \stex_deactivate_macro:Nn \renamedec1 {copymodules}
3422 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3423
3424
3425 \seq_new:N \l_stex_implicit_morphisms_seq
3426 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3427   \stex_import_module_uri:nn { #1 } { #2 }

```

```

3428 \stex_debug:nn{implicits}{
3429   Implicit~morphism:~
3430   \l_stex_module_ns_str ? \l__stex_features_name_str
3431 }
3432 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3433   \l_stex_module_ns_str ? \l__stex_features_name_str
3434 }{
3435   \msg_error:nnn{stex}{error/conflictingmodules}{
3436     \l_stex_module_ns_str ? \l__stex_features_name_str
3437   }
3438 }
3439
3440 % TODO
3441
3442
3443
3444 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3445   \l_stex_module_ns_str ? \l__stex_features_name_str
3446 }
3447 }
3448

```

32.2 The feature environment

structural@feature

```

3449
3450 \NewDocumentEnvironment{structural@feature}{ m m m }{
3451   \stex_if_in_module:F {
3452     \msg_set:nnn{stex}{error/nomodule}{
3453       Structural~Feature~has~to~occur~in~a~module:\\
3454       Feature~#2~of~type~#1\\
3455       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3456     }
3457     \msg_error:nn{stex}{error/nomodule}
3458   }
3459
3460   \str_set:Nx \l_stex_module_name_str {
3461     \prop_item:Nn \l_stex_current_module_prop
3462     { name } / #2 - feature
3463   }
3464
3465   \str_set:Nx \l_stex_module_ns_str {
3466     \prop_item:Nn \l_stex_current_module_prop
3467     { ns }
3468   }
3469
3470
3471   \str_clear:N \l_tmpa_str
3472   \seq_clear:N \l_tmpa_seq
3473   \tl_clear:N \l_tmpa_tl
3474   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3475     origname = #2,
3476     name      = \l_stex_module_name_str ,
3477     ns        = \l_stex_module_ns_str ,

```

```

3478 imports = \exp_not:o { \l_tmpa_seq } ,
3479 constants = \exp_not:o { \l_tmpa_seq } ,
3480 content = \exp_not:o { \l_tmpa_tl } ,
3481 file = \exp_not:o { \g_stex_currentfile_seq } ,
3482 lang = \l_stex_module_lang_str ,
3483 sig = \l_tmpa_str ,
3484 meta = \l_tmpa_str ,
3485 feature = #1 ,
3486 }
3487
3488 \stex_if_smsmode:TF {
3489   \stex_smsmode_set_codes:
3490 } {
3491   \begin{stex_annotate_env}{ feature:#1 }{}
3492   \stex_annotate_invisible:nnn{header}{}{ #3 }
3493 }
3494 {}{
3495   \str_set:Nx \l_tmpa_str {
3496     c_stex_feature_
3497     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3498     \prop_item:Nn \l_stex_current_module_prop { name }
3499     _prop
3500   }
3501   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3502   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3503   \stex_if_smsmode:TF {
3504     \exp_args:Nx \stex_add_to_sms:n {
3505       \prop_gset_from_keyval:cn {
3506         c_stex_feature_
3507         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3508         \prop_item:Nn \l_stex_current_module_prop { name }
3509         _prop
3510       } {
3511         origname = #2,
3512         name = \prop_item:cn { \l_tmpa_str } { name } ,
3513         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3514         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3515         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3516         content = \prop_item:cn { \l_tmpa_str } { content } ,
3517         file = \prop_item:cn { \l_tmpa_str } { file } ,
3518         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3519         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3520         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3521         feature = \prop_item:cn { \l_tmpa_str } { feature }
3522       }
3523     }
3524   } {
3525     \end{stex_annotate_env}
3526   }
3527 }
3528

```

32.3 Features

structure

```

3529
3530 \prop_new:N \l_stex_all_structures_prop
3531
3532 \keys_define:nn { stex / features / structure } {
3533   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3534 }
3535
3536 \cs_new_protected:Nn \__stex_features_structure_args:n {
3537   \str_clear:N \l__stex_features_structure_name_str
3538   \keys_set:nn { stex / features / structure } { #1 }
3539 }
3540
3541 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3542 %   \__stex_features_structure_args:n { ##1 }
3543 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3544 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3545 %   }
3546 %} {
3547 %
3548 %}
3549
3550 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3551   \__stex_features_structure_args:n { #1 }
3552   \str_if_empty:NT \l__stex_features_structure_name_str {
3553     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3554   }
3555   \exp_args:Nnnx
3556   \begin{structural@feature}{ structure }
3557     { \l__stex_features_structure_name_str }{}
3558     \seq_clear:N \l_tmpa_seq
3559     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3560
3561   }{
3562     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3563     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3564     \str_set:Nx \l_tmpa_str {
3565       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3566       \prop_item:Nn \l_stex_current_module_prop { name }
3567     }
3568     \seq_map_inline:Nn \l_tmpa_seq {
3569       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3570     }
3571     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3572     \exp_args:Nnx
3573     \AddToHookNext { env / mathstructure / after }{
3574       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3575         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3576       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3577     \STEXexport {
3578       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3579       { \prop_item:Nn \l_stex_current_module_prop { origname } }

```



```

3580         {\l_tmpa_str}
3581         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3582         {#2}{\l_tmpa_str}
3583     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3584     % \prop_item:Nn \l_stex_current_module_prop { origname },
3585     % \l_tmpa_str
3586     % }
3587     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3588     % #2,\l_tmpa_str
3589     % }
3590     % \tl_set:cx { #2 } {
3591     % \stex_invoke_structure:n { \l_tmpa_str }
3592     % }
3593 }
3594
3595 \end{structural@feature}
3596 % \g_stex_last_feature_prop
3597 }

```

\instantiate

```

3598 \seq_new:N \l__stex_features_structure_field_seq
3599 \str_new:N \l__stex_features_structure_field_str
3600 \str_new:N \l__stex_features_structure_def_tl
3601 \prop_new:N \l__stex_features_structure_prop
3602 \NewDocumentCommand \instantiate { m O{} m }{
3603     \stex_smsmode_set_codes:
3604     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3605     \prop_set_eq:Nc \l__stex_features_structure_prop {
3606         c_stex_feature_\l_tmpa_str _prop
3607     }
3608     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3609     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3610         \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3611         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3612             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3613             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3614             {!} \l_tmpa_tl
3615             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3616                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3617                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3618                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3619             }{
3620                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3621                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3622                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3623                 \l_tmpa_tl
3624                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3625                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3626                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3627                 }{
3628                     \tl_clear:N \l_tmpb_tl
3629                 }
3630             }
3631         }{

```

```

3632 \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3633 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3634   \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3635   \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3636   \tl_clear:N \l_tmpa_tl
3637 }{
3638   % TODO throw error
3639 }
3640 }
3641 % \l_tmpa_str: name
3642 % \l_tmpa_tl: definiens
3643 % \l_tmpb_tl: notation
3644 \tl_if_empty:NT \l__stex_features_structure_field_str {
3645   % TODO throw error
3646 }
3647 \str_clear:N \l_tmpb_str
3648
3649 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3650 \seq_map_inline:Nn \l_tmpa_seq {
3651   \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3652   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3653   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3654     \seq_map_break:n {
3655       \str_set:Nn \l_tmpb_str { ###1 }
3656     }
3657   }
3658 }
3659 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3660 \l_tmpb_str
3661
3662 \tl_if_empty:NTF \l_tmpb_tl {
3663   \tl_if_empty:NF \l_tmpa_tl {
3664     \exp_args:Nx \use:n {
3665       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3666     }
3667   }
3668 }{
3669   \tl_if_empty:NTF \l_tmpa_tl {
3670     \exp_args:Nx \use:n {
3671       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3672     }
3673   }
3674 }{
3675   \exp_args:Nx \use:n {
3676     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3677     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3678   }
3679 }
3680 }
3681 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3682 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3683 % #3/\l__stex_features_structure_field_str
3684 % \par
3685 % \expandafter\present\csname

```

```

3686 %      l_stex_symdecl_
3687 %      \prop_item:Nn \l_stex_current_module_prop {ns} ?
3688 %      \prop_item:Nn \l_stex_current_module_prop {name} ?
3689 %      #3/\l__stex_features_structure_field_str
3690 %      _prop
3691 %      \endcsname
3692 }
3693
3694 \tl_clear:N \l__stex_features_structure_def_tl
3695
3696 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3697 \seq_map_inline:Nn \l_tmpa_seq {
3698   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3699   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3700   \exp_args:Nx \use:n {
3701     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3702
3703     }
3704   }
3705
3706   \prop_if_exist:cF {
3707     l_stex_symdecl_
3708     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3709     \prop_item:Nn \l_stex_current_module_prop {name} ?
3710     #3/\l_tmpa_str
3711     _prop
3712   }{
3713     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3714     \l_tmpb_str
3715     \exp_args:Nx \use:n {
3716       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3717     }
3718   }
3719 }
3720
3721 \symdecl*[type={\STEXsymbol{module-type}}{
3722   \_stex_term_math_oms:nnnn {
3723     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3724     \prop_item:Nn \l__stex_features_structure_prop {name}
3725     }{}{0}{}
3726   }{}{#3}
3727
3728 % TODO: -> sms file
3729
3730 \tl_set:cx{ #3 }{
3731   \stex_invoke_structure:nnn {
3732     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3733     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3734   } {
3735     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3736     \prop_item:Nn \l__stex_features_structure_prop {name}
3737   }
3738 }
3739

```

3740 }

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

\stex_invoke_structure:nnn

```

3741 % #1: URI of the instance
3742 % #2: URI of the instantiated module
3743 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3744   \tl_if_empty:nTF{ #3 }{
3745     \prop_set_eq:Nc \l__stex_features_structure_prop {
3746       c_stex_feature_ #2 _prop
3747     }
3748     \tl_clear:N \l_tmpa_tl
3749     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3750     \seq_map_inline:Nn \l_tmpa_seq {
3751       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3752       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3753       \cs_if_exist:cT {
3754         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3755       }{
3756         \tl_if_empty:NF \l_tmpa_tl {
3757           \tl_put_right:Nn \l_tmpa_tl {,}
3758         }
3759         \tl_put_right:Nx \l_tmpa_tl {
3760           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3761         }
3762       }
3763     }
3764     \exp_args:No \mathstrut \l_tmpa_tl
3765   }{
3766     \stex_invoke_symbol:n{#1/#3}
3767   }
3768 }
```

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

3769 </package>

Chapter 33

STEX -Statements Implementation

```
3770 <*package>
3771
3772 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3773
3774 \protected\def\ignorespacesandpars{
3775   \begingroup\catcode13=10\relax
3776   \@ifnextchar\par{
3777     \endgroup\expandafter\ignorespacesandpars\@gobble
3778   }{
3779     \endgroup
3780   }
3781 }
3782
3783 <@@=stex_statements>
3784
3785   Warnings and error messages
```

\titleemph

```
3785 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3786 \keys_define:nn {stex / definiendum }{
3787   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3788   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3789   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3790 }
3791 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3792   \str_clear:N \l__stex_statements_definiendum_root_str
3793   \tl_clear:N \l__stex_statements_definiendum_post_tl
3794   \str_clear:N \l__stex_statements_definiendum_gfa_str
}
```

```

3795 \keys_set:nn { stex / definiendum } { #1 }
3796 }
3797 \NewDocumentCommand \definiendum { 0{} m m } {
3798   \__stex_statements_definiendum_args:n { #1 }
3799   \stex_get_symbol:n { #2 }
3800   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3801   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3802     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3803       \tl_set:Nn \l_tmpa_tl { #3 }
3804     } {
3805       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3806       \tl_set:Nn \l_tmpa_tl {
3807         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3808       }
3809     }
3810   } {
3811     \tl_set:Nn \l_tmpa_tl { #3 }
3812   }
3813
3814   % TODO root
3815   \rustex_if:TF {
3816     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3817   } {
3818     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3819   }
3820 }
3821 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3822 \NewDocumentCommand \definame { 0{} m } {
3823   \__stex_statements_definiendum_args:n { #1 }
3824   % TODO: root
3825   \stex_get_symbol:n { #2 }
3826   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3827   \str_set:Nx \l_tmpa_str {
3828     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3829   }
3830   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3831   \rustex_if:TF {
3832     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3833       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3834     }
3835   } {
3836     \defemph@uri {
3837       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3838     } { \l_stex_get_symbol_uri_str }
3839   }
3840 }
3841 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3842
3843 \keys_define:nn {stex / sdefinition }{
3844   type      .str_set_x:N = \sdefinitiontype,
3845   id        .str_set_x:N = \sdefinitionid,
3846   title     .tl_set:N     = \sdefinitiontitle
3847 }
3848 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3849   \str_clear:N \sdefinitiontype
3850   \str_clear:N \sdefinitionid
3851   \tl_clear:N \sdefinitiontitle
3852   \keys_set:nn { stex / sdefinition }{ #1 }
3853 }
3854
3855 \NewDocumentEnvironment{sdefinition}{0{}}{
3856   \__stex_statements_sdefinition_args:n{ #1 }
3857   \stex_reactivate_macro:N \definiendum
3858   \stex_reactivate_macro:N \definame
3859   \stex_smsmode_set_codes:
3860   \stex_if_smsmode:F {
3861     \exp_args:Nnnx
3862     \begin{stex_annotate_env}{definition}{}
3863     \str_if_empty:NF \sdefinitiontype {
3864       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3865     }
3866   }
3867   \clist_set:No \l_tmpa_clist \sdefinitiontype
3868   \tl_clear:N \l_tmpa_tl
3869   \clist_map_inline:Nn \l_tmpa_clist {
3870     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3871       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3872     }
3873   }
3874   \tl_if_empty:NTF \l_tmpa_tl {
3875     \__stex_statements_sdefinition_start:
3876   }{
3877     \l_tmpa_tl
3878   }
3879   \stex_ref_new_doc_target:n \sdefinitionid
3880 }{
3881   \clist_set:No \l_tmpa_clist \sdefinitiontype
3882   \tl_clear:N \l_tmpa_tl
3883   \clist_map_inline:Nn \l_tmpa_clist {
3884     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3885       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3886     }
3887   }
3888   \tl_if_empty:NTF \l_tmpa_tl {
3889     \__stex_statements_sdefinition_end:
3890   }{
3891     \l_tmpa_tl
3892   }
3893   \stex_if_smsmode:F {
3894     \end{stex_annotate_env}

```

```

3895 }
3896 }

```

`\stexpatchdefinition`

```

3897 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3898   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3899     ~(\sdefinitiontitle)
3900   }~}
3901 }
3902 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3903
3904 \newcommand\stexpatchdefinition[3] [] {
3905   \str_set:Nx \l_tmpa_str{ #1 }
3906   \str_if_empty:NTF \l_tmpa_str {
3907     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3908     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3909   }{
3910     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3911     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3912   }
3913 }

```

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

`\inlinedef inline:`

```

3914 \NewDocumentCommand \inlinedef { m } {
3915   \begingroup
3916   \stex_reactivate_macro:N \definiendum
3917   \stex_reactivate_macro:N \definame
3918   \stex_ref_new_doc_target:n{
3919     #1
3920   }
3921 }

```

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

33.2 Assertions

`sassertion`

```

3922
3923 \keys_define:nn {stex / sassertion }{
3924   type      .str_set_x:N = \sassertiontype,
3925   id        .str_set_x:N = \sassertionid,
3926   title     .tl_set:N     = \sassertiontitle ,
3927   name      .str_set_x:N = \sassertionname
3928 }
3929 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3930   \str_clear:N \sassertiontype
3931   \str_clear:N \sassertionid
3932   \str_clear:N \sassertionname
3933   \tl_clear:N \sassertiontitle
3934   \keys_set:nn { stex / sassertion }{ #1 }
3935 }

```



```

3936
3937 %\tl_new:N \g__stex_statements_aftergroup_tl
3938
3939 \NewDocumentEnvironment{sassertion}{0{}}{
3940   \__stex_statements_sassertion_args:n{ #1 }
3941   \stex_smsmode_set_codes:
3942   \stex_if_smsmode:F {
3943     \exp_args:Nnnx
3944     \begin{stex_annotate_env}{assertion}{}
3945     \str_if_empty:NF \sassertiontype {
3946       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3947     }
3948   }
3949   \clist_set:No \l_tmpa_clist \sassertiontype
3950   \tl_clear:N \l_tmpa_tl
3951   \clist_map_inline:Nn \l_tmpa_clist {
3952     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3953       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3954     }
3955   }
3956   \tl_if_empty:NTF \l_tmpa_tl {
3957     \__stex_statements_sassertion_start:
3958   }{
3959     \l_tmpa_tl
3960   }
3961   \stex_ref_new_doc_target:n \sassertionid
3962 }{
3963   \clist_set:No \l_tmpa_clist \sassertiontype
3964   \tl_clear:N \l_tmpa_tl
3965   \clist_map_inline:Nn \l_tmpa_clist {
3966     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3967       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3968     }
3969   }
3970   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
3971   \tl_if_empty:NTF \l_tmpa_tl {
3972     \__stex_statements_sassertion_end:
3973   }{
3974     \l_tmpa_tl
3975   }
3976   \stex_if_smsmode:F {
3977     \end{stex_annotate_env}
3978   }
3979 }

```

\stexpatchassertion

```

3980
3981 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3982   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
3983     (\sassertiontitle)
3984   }~}
3985 }
3986 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3987

```

```

3988 \newcommand\stexpatchassertion[3] [] {
3989   \str_set:Nx \l_tmpa_str{ #1 }
3990   \str_if_empty:NTF \l_tmpa_str {
3991     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3992     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3993   }{
3994     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3995     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3996   }
3997 }

```

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

\inlineass inline:

```

3998 \NewDocumentCommand \inlineass { m } {
3999   \begingroup
4000   \stex_ref_new_doc_target:n{
4001     #1
4002   \endgroup
4003 }

```

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

33.3 Examples

sexample

```

4004
4005 \keys_define:nn {stex / sexample }{
4006   type      .str_set_x:N = \exampletype,
4007   id        .str_set_x:N = \sexampleid,
4008   title     .tl_set:N     = \sexampletitle,
4009   for       .clist_set:N  = \sexamplefor,
4010 }
4011 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4012   \str_clear:N \sexampletype
4013   \str_clear:N \sexampleid
4014   \tl_clear:N \sexampletitle
4015   \clist_clear:N \sexamplefor
4016   \keys_set:nn { stex / sexample }{ #1 }
4017 }
4018
4019 \NewDocumentEnvironment{sexample}{0{}}{
4020   \__stex_statements_sexample_args:n{ #1 }
4021   \stex_smsmode_set_codes:
4022   \stex_if_smsmode:F {
4023     \seq_clear:N \l_tmpa_seq
4024     \clist_map_inline:Nn \sexamplefor {
4025       \str_if_eq:nnF{ ##1 }{}{
4026         \stex_get_symbol:n { ##1 }
4027         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4028           \l_stex_get_symbol_uri_str
4029         }
4030       }

```

```

4031 }
4032 \exp_args:Nnnx
4033 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4034 \str_if_empty:NF \sexamplotype {
4035   \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4036 }
4037 }
4038 \stex_ref_new_doc_target:n \sexampleid
4039 \clist_set:No \l_tmpa_clist \sexamplotype
4040 \tl_clear:N \l_tmpa_tl
4041 \clist_map_inline:Nn \l_tmpa_clist {
4042   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4043     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4044   }
4045 }
4046 \tl_if_empty:NTF \l_tmpa_tl {
4047   \__stex_statements_sexample_start:
4048 }{
4049   \l_tmpa_tl
4050 }
4051 }{
4052   \clist_set:No \l_tmpa_clist \sexamplotype
4053   \tl_clear:N \l_tmpa_tl
4054   \clist_map_inline:Nn \l_tmpa_clist {
4055     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4056       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4057     }
4058   }
4059   \tl_if_empty:NTF \l_tmpa_tl {
4060     \__stex_statements_sexample_end:
4061   }{
4062     \l_tmpa_tl
4063   }
4064   \stex_if_smsmode:F {
4065     \end{stex_annotate_env}
4066   }
4067 }

```

\stexpatchexample

```

4068
4069 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4070   \par\noindent\titl{Example~\tl_if_empty:NF \sexamplotype {
4071     (\sexamplotype)
4072   }~}
4073 }
4074 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4075
4076 \newcommand\stexpatchexample[3]{} {
4077   \str_set:Nx \l_tmpa_str{ #1 }
4078   \str_if_empty:NTF \l_tmpa_str {
4079     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4080     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4081   }{
4082     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }

```

```

4083     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4084   }
4085 }

```

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

`\inlineex` inline:

```

4086 \NewDocumentCommand \inlineex { m } {
4087   \begingroup
4088   \stex_ref_new_doc_target:n{
4089     #1
4090   \endgroup
4091 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4092 \keys_define:nn { stex / sparagraph } {
4093   id      .str_set_x:N = \sparagraphid ,
4094   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4095   type    .str_set_x:N = \sparagraphtype ,
4096   for     .str_set_x:N = \sparagraphfor ,
4097   from    .tl_set_x:N  = \sparagraphfrom ,
4098   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4099   name    .str_set:N   = \sparagraphname
4100 }
4101
4102 \cs_new_protected:Nn \stex_sparagraph_args:n {
4103   \tl_clear:N \l_stex_sparagraph_title_tl
4104   \tl_clear:N \sparagraphfrom
4105   \tl_clear:N \l_stex_sparagraph_start_tl
4106   \str_clear:N \sparagraphid
4107   \str_clear:N \sparagraphtype
4108   \str_clear:N \sparagraphfor
4109   \str_clear:N \sparagraphname
4110   \keys_set:nn { stex / sparagraph }{ #1 }
4111 }
4112 \newif\if@in@omtext\@in@omtextfalse
4113
4114 \NewDocumentEnvironment {sparagraph} { 0{} } {
4115   \stex_sparagraph_args:n { #1 }
4116   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4117     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4118   }{
4119     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4120   }
4121   \@in@omtexttrue
4122   \stex_smsmode_set_codes:
4123   \stex_if_smsmode:F {
4124     \exp_args:Nnnx
4125     \begin{stex_annotate_env}{paragraph}{}

```

```

4126 \str_if_empty:NF \sparagraphtype {
4127 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4128 }
4129 }
4130 \clist_set:N \l_tmpa_clist \sparagraphtype
4131 \tl_clear:N \l_tmpa_tl
4132 \clist_map_inline:Nn \l_tmpa_clist {
4133 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4134 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4135 }
4136 }
4137 \tl_if_empty:NTF \l_tmpa_tl {
4138 \__stex_statements_sparagraph_start:
4139 }{
4140 \l_tmpa_tl
4141 }
4142 \stex_ref_new_doc_target:n \sparagraphid
4143 \ignorespacesandpars
4144 }{
4145 \clist_set:N \l_tmpa_clist \sparagraphtype
4146 \tl_clear:N \l_tmpa_tl
4147 \clist_map_inline:Nn \l_tmpa_clist {
4148 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4149 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4150 }
4151 }
4152 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4153 \tl_if_empty:NTF \l_tmpa_tl {
4154 \__stex_statements_sparagraph_end:
4155 }{
4156 \l_tmpa_tl
4157 }
4158 \stex_if_smsmode:F {
4159 \end{stex_annotate_env}
4160 }
4161 }

```

\stexpatchparagraph

```

4162
4163 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4164 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4165 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4166 \titleemph{\l_stex_sparagraph_title_tl}:~
4167 }
4168 }{
4169 \titleemph{\l_stex_sparagraph_start_tl}~
4170 }
4171 }
4172 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4173
4174 \newcommand\stexpatchparagraph[3] [] {
4175 \str_set:Nx \l_tmpa_str{ #1 }
4176 \str_if_empty:NTF \l_tmpa_str {
4177 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }

```

```

4178     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4179   }{
4180     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4181     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4182   }
4183 }

```

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

symboldoc

```

4184 \NewDocumentEnvironment{symboldoc}{ m }{
4185   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4186   \seq_clear:N \l_tmpb_seq
4187   \seq_map_inline:Nn \l_tmpa_seq {
4188     \str_if_eq:nnF{ ##1 }{}{
4189       \stex_get_symbol:n { ##1 }
4190       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4191         \l_stex_get_symbol_uri_str
4192       }
4193     }
4194   }
4195   \par
4196   \exp_args:Nnnx
4197   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4198 }{
4199   \end{stex_annotate_env}
4200 }
4201 \</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

```
4202 <*package>
4203 <@@=stex_sproof>
4204
4205 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4206
```

34.2 Proofs

We first define some keys for the proof environment.

```
4207 \keys_define:nn { stex / spf } {
4208   id          .str_set:N = \l__stex_sproof_spf_id_str,
4209   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4210   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4211   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4212   proofend   .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4213   type       .tl_set:N  = \l__stex_sproof_spf_type_tl,
4214   title      .tl_set:N  = \l__stex_sproof_spf_title_tl,
4215   continues  .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4216   functions  .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4217   method     .tl_set:N  = \l__stex_sproof_spf_method_tl
4218 }
4219 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4220   \str_clear:N \l__stex_sproof_spf_id_str
4221   \tl_clear:N \l__stex_sproof_spf_display_tl
4222   \tl_clear:N \l__stex_sproof_spf_for_tl
4223   \tl_clear:N \l__stex_sproof_spf_from_tl
4224   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4225   \tl_clear:N \l__stex_sproof_spf_type_tl
4226   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4227 \tl_clear:N \l__stex_sproof_spf_continues_tl
4228 \tl_clear:N \l__stex_sproof_spf_functions_tl
4229 \tl_clear:N \l__stex_sproof_spf_method_tl
4230 \keys_set:nn { stex / spf }{ #1 }
4231 }

```

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

```

4232 \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.

```

4233 \newcount\count_ten
4234 \newenvironment{pst@with@label}[1]{
4235   \edef\pst@label{#1}
4236   \advance\count_ten by 1\relax
4237   \count_ten=1
4238 }{
4239   \advance\count_ten by -1\relax
4240 }

```

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

```

4241 \def\the@pst@label{
4242   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4243 }

```

(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.

```

4244 \keys_define:nn { stex / pstlabel }{
4245   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4246   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4247   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4248 }
4249 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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


```

4250 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4251 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4252 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4253 }
4254 \__stex_sproof_pstlabel_args:n {}
4255 \newcommand\setpstlabelstyle[1]{
4256   \__stex_sproof_pstlabel_args:n {#1}
4257 }
4258 \newcommand\setpstlabelstyledefault{%
4259   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4260 }

```

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

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

```

4261 \ExplSyntaxOff
4262 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4263 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4264 \def\pst@make@label@short#1#2{#2}
4265 \def\pst@make@label@empty#1#2{}
4266 \ExplSyntaxOn
4267 \def\pstlabelstyle#1{%
4268   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4269 }%
4270 \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.

```

4271 \def\next@pst@label{%
4272   \global\advance\count\count10 by 1%
4273 }%

```

(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

```

4274 \def\sproof@box{
4275   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4276 }
4277 \def\spf@proofend{\sproof@box}
4278 \def\sproofend{
4279   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4280     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4281   }
4282 }
4283 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4284 \def\spf@proofsketch@kw{Proof Sketch}
4285 \def\spf@proof@kw{Proof}
4286 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4287 \AddToHook{begindocument}{
4288   \ltx@ifpackageloaded{babel}{
4289     \makeatletter
4290     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4291     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4292       \input{sproof-ngerman.ldf}
4293     }
4294     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4295       \input{sproof-finnish.ldf}
4296     }
4297     \clist_if_in:NnT \l_tmpa_clist {french}{
4298       \input{sproof-french.ldf}
4299     }
4300     \clist_if_in:NnT \l_tmpa_clist {russian}{
4301       \input{sproof-russian.ldf}
4302     }
4303     \makeatother
4304   }
4305 }
```

`spfsketch`

```

4306 \newcommand\spfsketch[2][]{
4307   \__stex_sproof_spf_args:n{#1}
4308   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4309     \titleemph{
4310       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4311         \spf@proofsketch@kw
4312       }{
4313         \l__stex_sproof_spf_type_tl
4314       }
4315     }:
4316   }
4317   {~#2}
4318   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4319   \sproofend
4320 }
```

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

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

```

4321 \newenvironment{spfeq}[2][]{
4322   \__stex_sproof_spf_args:n{#1}
4323   %\sref@target
4324   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4325     \titleemph{
4326       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4327         \spf@proof@kw
4328       }{

```

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

¹⁵EDNOTE: document above

```

4329     \l__stex_sproof_spf_type_tl
4330   }
4331   }:
4332 }
4333 {-#2}
4334 \begin{displaymath}\begin{array}{rcll}
4335 }{
4336 \end{array}\end{displaymath}
4337 }

```

(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.

```

4338 \newenvironment{spf@proof}[2] []{
4339   \l__stex_sproof_spf_args:n{#1}
4340   %\sref@target
4341   \count_ten=10
4342   \par\noindent
4343   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4344     \titleemph{
4345       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4346         \spf@proof@kw
4347       }{
4348         \l__stex_sproof_spf_type_tl
4349       }
4350     }:
4351   }
4352   {-#2}
4353   %\sref@label{id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4354   \def\pst@label{}
4355   \newcount\pst@count% initialize the labeling mechanism
4356   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4357   }{
4358     \end{pst@with@label}\end{description}
4359   }
4360   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4361   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4362 \newcommand\spfidea[2] []{
4363   \l__stex_sproof_spf_args:n{#1}
4364   \titleemph{
4365     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4366       \l__stex_sproof_spf_type_tl
4367     }:
4368   }-#2
4369   \sproofend
4370 }

```

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

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

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

EdN:16

```

spfstep 16
4371 \newenvironment{spfstep}[1][]{
4372   \_stex_sproof_spf_args:n{#1}
4373   \@in@omtexttrue
4374   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4375     \item[\the@pst@label]
4376   }
4377   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4378     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4379   }
4380   %\sref@label{id{\pst@label}
4381   \ignorespacesandpars
4382 }{
4383   \next@pst@label\ignorespacesandpars
4384 }

```

sproofcomment

```

4385 \newenvironment{sproofcomment}[1][]{
4386   \_stex_sproof_spf_args:n{#1}
4387   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4388     \item[\the@pst@label]
4389   }
4390 }{
4391   \next@pst@label
4392 }

```

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.

```

4393 \newenvironment{subproof}[2][]{
4394   \_stex_sproof_spf_args:n{#1}
4395   \def\@test{#2}
4396   \ifx\@test\empty\else
4397     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4398       \item[\the@pst@label]
4399     }{#2}
4400   \fi
4401   \begin{pst@with@label}{\pst@label,\number\count_ten}
4402 }{
4403   \end{pst@with@label}\next@pst@label
4404 }

```

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

```

4405 \newenvironment{spfcases}[2][]{
4406   \def\@test{#1}
4407   \ifx\@test\empty
4408     \begin{subproof}[method=by-cases]{#2}

```

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

```

4409 \else
4410   \begin{subproof}[#1,method=by-cases]{#2}
4411 \fi
4412 }{
4413   \end{subproof}
4414 }

```

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

```

4415 \newenvironment{spfcase}[2] [] {
4416   \__stex_sproof_spf_args:n{#1}
4417   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4418     \item[\the@pst@label]
4419   }
4420   \def\@test{#2}
4421   \ifx\@test\@empty
4422   \else
4423     {\titleemph{#2}:~}
4424   \fi
4425   \begin{pst@with@label}{\pst@label,\number\count_ten}
4426 }{
4427   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4428     \sproofend
4429   }
4430   \end{pst@with@label}
4431   \next@pst@label
4432 }

```

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

```

4433 \newcommand\spfcasesketch[3] [] {
4434   \__stex_sproof_spf_args:n{#1}
4435   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4436     \item[\the@pst@label]
4437   }
4438   \def\@test{#2}
4439   \ifx\@test\@empty
4440   \else
4441     {\titleemph{#2}:~}
4442   \fi#3
4443   \next@pst@label
4444 }%

```

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.

```

4445 \keys_define:nn { stex / just }{
4446   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4447   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4448   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4449   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4450 }

```

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

`justification`

```
4451 \newenvironment{justification}[1] [] {}{}
```

`\premise`

```
4452 \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.

```
4453 \newcommand\justarg[2] [] {#2}
```

```
4454 \end{package}
```

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

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

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

Chapter 35

STEX -Others Implementation

```
4455 <*package>
4456
4457 %%%%%%%%%% others.dtx %%%%%%%%%%
4458
4459 <@@=stex_others>
    Warnings and error messages
4460 % None

\MSC Math subject classifier

4461 \NewDocumentCommand \MSC {m} {
4462 % TODO
4463 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4464 \@ifpackageloaded{tikzinput}{
4465 \RequirePackage{stex-tikzinput}
4466 }{}
4467 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4468 \*package>
4469 \@@=stex_modules>
4470
4471 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4472
4473 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4474 \begingroup
4475 \stex_module_setup:nn{
4476   ns=\c_stex_metatheory_ns_str,
4477   meta=NONE
4478 }{Metatheory}
4479 \stex_reactivate_macro:N \symdecl
4480 \stex_reactivate_macro:N \notation
4481 \stex_reactivate_macro:N \symdef
4482 \ExplSyntaxOff
4483 \csname stex_suppress_html:n\endcsname{
4484   % is-a (a:A, a \in A, a is an A, etc.)
4485   \symdecl[args=ai]{isa}
4486   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4487   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4488   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4489
4490   % bind (\forall, \Pi, \lambda etc.)
4491   \symdecl[args=Bi]{bind}
4492   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4493   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4494   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4495
4496   % dummy variable
4497   \symdecl{dummyvar}
4498   \notation[underscore]{dummyvar}{\comp\_}
4499   \notation[dot]{dummyvar}{\comp\cdot}
4500   \notation[dash]{dummyvar}{\comp{\rm --}}
4501
4502   %fromto (function space, Hom-set, implication etc.)
```



```

4503 \symdecl[args=ai]{fromto}
4504 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4505 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4506
4507 % mapto (lambda etc.)
4508 %\symdecl[args=Bi]{mapto}
4509 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4510 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4511 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4512
4513 % function/operator application
4514 \symdecl[args=ia]{apply}
4515 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4516 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4517
4518 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4519 \symdecl{collection}
4520 \notation[U]{collection}{\comp{\mathcal{U}}}
4521 \notation[set]{collection}{\comp{\textsf{Set}}}
4522
4523 % sequences
4524 \symdecl[args=1]{seqtype}
4525 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4526
4527 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4528 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4529
4530 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4531 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4532 % ^ superceded by \aseqfromto and \livar/\uivar
4533
4534 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4535 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4536 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4537
4538 % letin (‘‘let’’, local definitions, variable substitution)
4539 \symdecl[args=bii]{letin}
4540 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4541 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4542 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4543
4544 % structures
4545 \symdecl*[args=1]{module-type}
4546 \notation{module-type}{\mathtt{MOD} #1}
4547 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4548 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4549
4550 }
4551 \ExplSyntaxOn
4552 \stex_add_to_current_module:n{
4553   \let\nappa\apply
4554   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4555   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4556   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4557 \def\uivar{\csname sequence-index\endcsname[ui]}
4558 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4559 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4560 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4561 }
4562 \__stex_modules_end_module:
4563 \endgroup
4564 \endpackage

```

Chapter 37

Tikzinput Implementation

```
4565 <*package>
4566
4567 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4568
4569 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4570 \RequirePackage{l3keys2e}
4571
4572 \keys_define:nn { tikzinput } {
4573   image .bool_set:N = \c_tikzinput_image_bool,
4574   image .default:n = false ,
4575   unknown .code:n = {}
4576 }
4577
4578 \ProcessKeysOptions { tikzinput }
4579
4580 \bool_if:NTF \c_tikzinput_image_bool {
4581   \RequirePackage{graphicx}
4582
4583   \providecommand\usetikzlibrary[]{}
4584   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4585 }{
4586   \RequirePackage{tikz}
4587   \RequirePackage{standalone}
4588
4589   \newcommand \tikzinput [2] [] {
4590     \setkeys{Gin}{#1}
4591     \ifx \Gin@ewidth \Gin@exclamation
4592       \ifx \Gin@eheight \Gin@exclamation
4593         \input { #2 }
4594       \else
4595         \resizebox{!}{ \Gin@eheight }{
4596           \input { #2 }
4597         }
4598       \fi
4599     \else
4600       \ifx \Gin@eheight \Gin@exclamation
4601         \resizebox{ \Gin@ewidth }{!}{
4602           \input { #2 }
```

```

4603     }
4604     \else
4605         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4606             \input { #2 }
4607         }
4608     \fi
4609 \fi
4610 }
4611 }
4612
4613 \newcommand \ctikzinput [2] [] {
4614     \begin{center}
4615         \tikzinput [1] {#2}
4616     \end{center}
4617 }
4618
4619 \@ifpackageloaded{stex}{
4620     \RequirePackage{stex-tikzinput}
4621 }{}
4622
4623 </package>
4624 <*stex>
4625 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4626 \RequirePackage{stex}
4627 \RequirePackage{tikzinput}
4628
4629 \newcommand\mhtikzinput [2] [] {%
4630     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4631     \stex_in_repository:nn\Gin@mhrepos{
4632         \tikzinput [1]{\mhpath{##1}{#2}}
4633     }
4634 }
4635 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4636 </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.

```
4637 \*cls)
4638 \@@=document_structure)
4639 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4640 \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`

```
4641 \keys_define:nn{ document-structure / pkg }{
4642   class      .str_set_x:N = \c_document_structure_class_str,
4643   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4644   report     .code:n      = {
4645     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4646     \str_set:Nn \c_document_structure_class_str {report}
4647   },
4648   book       .code:n      = {
4649     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4650     \str_set:Nn \c_document_structure_class_str {book}
4651   },
4652   bookpart   .code:n      = {
4653     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4654     \str_set:Nn \c_document_structure_class_str {book}
4655     \str_set:Nn \c_document_structure_topsect_str {chapter}
4656   },
```

```

4657 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4658 unknown     .code:n      = {
4659   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4660 }
4661 }
4662 \ProcessKeysOptions{ document-structure / pkg }
4663 \str_if_empty:NT \c_document_structure_class_str {
4664   \str_set:Nn \c_document_structure_class_str {article}
4665 }
4666 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4667   {\c_document_structure_class_str}
4668

```

38.3 Beefing up the document environment

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

```

4669 \RequirePackage{omdoc}
4670 \bool_if:NF \c_document_structure_minimal_bool {
4671   \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

```

4672 \keys_define:nn { document-structure / document }{
4673   id .str_set_x:N = \c_document_structure_document_id_str
4674 }
4675 \let\__document_structure_orig_document=\document
4676 \renewcommand{\document}[1][]{
4677   \keys_set:nn{ document-structure / document }{ #1 }
4678   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4679   \__document_structure_orig_document
4680 }

```

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

```

4681 }
4682 \</cls>

```

38.4 Implementation: OMDoc Package

```

4683 \<*package>
4684 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4685 \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

```

4686
4687 \keys_define:nn{ document-structure / pkg }{
4688   class      .str_set_x:N = \c_document_structure_class_str,
4689   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4690   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4691 }
4692 \ProcessKeysOptions{ document-structure / pkg }
4693 \str_if_empty:NT \c_document_structure_class_str {
4694   \str_set:Nn \c_document_structure_class_str {article}
4695 }
4696 \str_if_empty:NT \c_document_structure_topsect_str {
4697   \str_set:Nn \c_document_structure_topsect_str {section}
4698 }

```

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

```

4699 \RequirePackage{xspace}
4700 \RequirePackage{comment}
4701 \AddToHook{begindocument}{
4702   \ltx@ifpackageloaded{babel}{
4703     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4704     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4705       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4706     }
4707   }{}
4708 }

```

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

```

4709 %\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.

```

4710 \int_new:N \l_document_structure_section_level_int
4711 \str_case:NnF \c_document_structure_topsect_str {
4712   {part}{
4713     \int_set:Nn \l_document_structure_section_level_int {0}
4714   }
4715   {chapter}{
4716     \int_set:Nn \l_document_structure_section_level_int {1}
4717   }
4718 }{
4719   \str_case:NnF \c_document_structure_class_str {
4720     {book}{
4721       \int_set:Nn \l_document_structure_section_level_int {0}
4722     }
4723     {report}{
4724       \int_set:Nn \l_document_structure_section_level_int {0}
4725     }
4726   }{
4727     \int_set:Nn \l_document_structure_section_level_int {2}
4728   }
4729 }

```

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

```
4730 \def\current@section@level{document}%
4731 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4732 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4733 \cs_new_protected:Npn \skipomgroup {
4734   \ifcase\l_document_structure_section_level_int
4735   \or\stepcounter{part}
4736   \or\stepcounter{chapter}
4737   \or\stepcounter{section}
4738   \or\stepcounter{subsection}
4739   \or\stepcounter{subsubsection}
4740   \or\stepcounter{paragraph}
4741   \or\stepcounter{subparagraph}
4742   \fi
4743 }
```

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

`blindomgroup`

```
4744 \newcommand\at@begin@blindomgroup[1]{%
4745 \newenvironment{blindomgroup}
4746 {
4747   \int_incr:N\l_document_structure_section_level_int
4748   \at@begin@blindomgroup\l_document_structure_section_level_int
4749 }{}}
```

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

```
4750 \newcommand\omgroup@nonum[2]{
4751   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4752   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4753 }
```

(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.

```
4754 \newcommand\omgroup@num[2]{
```

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


```

4755 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4756   \@nameuse{#1}{#2}
4757 }{
4758   \cs_if_exist:NTF\rdfmata@sectioning{
4759     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4760   }{
4761     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4762   }
4763 }
4764 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4765 }

```

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

omgroup

```

4766 \keys_define:nn { document-structure / omgroup }{
4767   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4768   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4769   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4770   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4771   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4772   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4773   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4774   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4775   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4776   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4777 }
4778 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4779   \str_clear:N \l__document_structure_omgroup_id_str
4780   \str_clear:N \l__document_structure_omgroup_date_str
4781   \clist_clear:N \l__document_structure_omgroup_creators_clist
4782   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4783   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4784   \tl_clear:N \l__document_structure_omgroup_type_tl
4785   \tl_clear:N \l__document_structure_omgroup_short_tl
4786   \tl_clear:N \l__document_structure_omgroup_display_tl
4787   \tl_clear:N \l__document_structure_omgroup_intro_tl
4788   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4789   \keys_set:nn { document-structure / omgroup } { #1 }
4790 }

```

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

```

4791 \newif\if@mainmatter\@mainmattertrue
4792 \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.

```

4793 \keys_define:nn { document-structure / sectioning }{
4794   name .str_set_x:N = \l__document_structure_sect_name_str ,
4795   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4796   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4797   num .bool_set:N = \l__document_structure_sect_num_bool ,
4798 }

```

```

4799 \cs_new_protected:Nn \l__document_structure_sect_args:n {
4800   \str_clear:N \l__document_structure_sect_name_str
4801   \str_clear:N \l__document_structure_sect_ref_str
4802   \bool_set_false:N \l__document_structure_sect_clear_bool
4803   \bool_set_false:N \l__document_structure_sect_num_bool
4804   \keys_set:nn { document-structure / sectioning } { #1 }
4805 }
4806 \newcommand\omdoc@sectioning[3][]{
4807   \l__document_structure_sect_args:n {#1}
4808   \let\omdoc@sect@name\l__document_structure_sect_name_str
4809   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4810   \if@mainmatter% numbering not overridden by frontmatter, etc.
4811     \bool_if:NTF \l__document_structure_sect_num_bool {
4812       \omgroup@num{#2}{#3}
4813     }{
4814       \omgroup@nonum{#2}{#3}
4815     }
4816     \def\current@section@level{\omdoc@sect@name}
4817   \else
4818     \omgroup@nonum{#2}{#3}
4819   \fi
4820 }% 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.

```

4821 \newcommand\omgroup@redefine@addtocontents[1]{%
4822   %\edef\__document_structureimport{#1}%
4823   %\@for\@I:=\__document_structureimport\do{%
4824     %\edef\@path{\csname module@\@I @path\endcsname}%
4825     %\@ifundefined{tf@toc}\relax%
4826     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4827   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4828   %\def\addcontentsline##1##2##3{%
4829     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4830   %\else% hyperref.sty not loaded
4831   %\def\addcontentsline##1##2##3{%
4832     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4833   %\fi
4834   }% 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.

```

4835 \int_new:N \l__document_structure_omgroup_level_int
4836 \newenvironment{omgroup}[2][]{% keys, title
4837 {
4838   \l__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.

```

4839 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4840   \omgroup@redefine@addtocontents{
4841     %\@ifundefined{module@id}\used@modules%
4842     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4843     }
4844 }

```

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

```

4845 \int_incr:N \l_document_structure_omgroup_level_int
4846 \int_incr:N \l_document_structure_section_level_int
4847 \ifcase\l_document_structure_section_level_int
4848   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4849   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4850   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4851   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4852   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4853   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4854   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4855 \fi
4856 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4857 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4858 }% for customization
4859 {}

```

and finally, we localize the sections

```

4860 \newcommand\omdoc@part@kw{Part}
4861 \newcommand\omdoc@chapter@kw{Chapter}
4862 \newcommand\omdoc@section@kw{Section}
4863 \newcommand\omdoc@subsection@kw{Subsection}
4864 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4865 \newcommand\omdoc@paragraph@kw{paragraph}
4866 \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`

```

4867 \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).

```

4868 \cs_if_exist:NTF\frontmatter{
4869   \let\__document_structure_orig_frontmatter\frontmatter
4870   \let\frontmatter\relax
4871 }{
4872   \tl_set:Nn\__document_structure_orig_frontmatter{
4873     \clearpage
4874     \@mainmatterfalse
4875     \pagenumbering{roman}
4876   }
4877 }
4878 \cs_if_exist:NTF\backmatter{

```

```

4879 \let\__document_structure_orig_backmatter\backmatter
4880 \let\backmatter\relax
4881 }{
4882 \tl_set:Nn\__document_structure_orig_backmatter{
4883 \clearpage
4884 \@mainmatterfalse
4885 \pagenumbering{roman}
4886 }
4887 }

```

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.

```

4888 \newenvironment{frontmatter}{
4889 \__document_structure_orig_frontmatter
4890 }{
4891 \cs_if_exist:NTF\mainmatter{
4892 \mainmatter
4893 }{
4894 \clearpage
4895 \@mainmattertrue
4896 \pagenumbering{arabic}
4897 }
4898 }

```

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

```

4899 \newenvironment{backmatter}{
4900 \__document_structure_orig_backmatter
4901 }{
4902 \cs_if_exist:NTF\mainmatter{
4903 \mainmatter
4904 }{
4905 \clearpage
4906 \@mainmattertrue
4907 \pagenumbering{arabic}
4908 }
4909 }

```

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

```

4910 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4911 \def \c__document_structure_document_str{document}
4912 \newcommand\afterprematurestop{}
4913 \def\prematurestop@endomgroup{
4914 \unless\ifx\@currenvir\c__document_structure_document_str
4915 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4916 \expandafter\prematurestop@endomgroup
4917 \fi
4918 }
4919 \providecommand\prematurestop{

```

```

4920 \message{Stopping~sTeX~processing~prematurely}
4921 \prematurestop@endomgroup
4922 \afterprematurestop
4923 \end{document}
4924 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

4925 \RequirePackage{etoolbox}
4926 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4927 \newrobustcmd\useSGvar[1]{%
4928   \@ifundefined{sTeX@Gvar@#1}
4929   {\PackageError{omdoc}
4930    {The sTeX Global variable #1 is undefined}
4931    {set it with \protect\setSGvar}}
4932   \@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.

```

4933 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4934   \@ifundefined{sTeX@Gvar@#1}
4935   {\PackageError{omdoc}
4936    {The sTeX Global variable #1 is undefined}
4937    {set it with \protect\setSGvar}}
4938   {\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.

```
4939 \*cls)
4940 \@@=mikoslides)
4941 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4942 \RequirePackage{l3keys2e,expl-keystr-compat}
4943
4944 \keys_define:nn{mikoslides / cls}{
4945   class .code:n = {
4946     \PassOptionsToClass{\CurrentOption}{omdoc}
4947     \str_if_eq:nnT{#1}{book}{
4948       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4949     }
4950     \str_if_eq:nnT{#1}{report}{
4951       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4952     }
4953   },
4954   notes .bool_set:N = \c__mikoslides_notes_bool ,
4955   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4956   unknown .code:n = {
4957     \PassOptionsToClass{\CurrentOption}{omdoc}
4958     \PassOptionsToClass{\CurrentOption}{beamer}
4959     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4960   }
4961 }
4962 \ProcessKeysOptions{ mikoslides / cls }
4963 \bool_if:NTF \c__mikoslides_notes_bool {
4964   \PassOptionsToPackage{notes=true}{mikoslides}
4965 }{
4966   \PassOptionsToPackage{notes=false}{mikoslides}
4967 }
4968 \</cls)
```

now we do the same for the mikoslides package.

```

4969 <*package>
4970 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4971 \RequirePackage{l3keys2e,expl-keystr-compat}
4972
4973 \keys_define:nn{mikoslides / pkg}{
4974   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4975   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4976   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4977   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4978   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4979   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4980   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4981   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4982   unknown      .code:n      = {
4983     \PassOptionsToClass{\CurrentOption}{stex}
4984     \PassOptionsToClass{\CurrentOption}{tikzinput}
4985   }
4986 }
4987 \ProcessKeysOptions{ mikoslides / pkg }
4988 \newif\ifnotes
4989 \bool_if:NTF \c__mikoslides_notes_bool {
4990   \notesttrue
4991 }{
4992   \notesfalse
4993 }
4994

```

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

```

4995 \str_if_empty:NTF \c__mikoslides_topsect_str {
4996   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4997 }{
4998   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4999 }
5000 </package>

```

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

```

5001 <*cls>
5002 \bool_if:NTF \c__mikoslides_notes_bool {
5003   \LoadClass{omdoc}
5004 }{
5005   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5006   \newcounter{Item}
5007   \newcounter{paragraph}
5008   \newcounter{subparagraph}
5009   \newcounter{Hfootnote}
5010   \RequirePackage{omdoc}
5011 }

```

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

```

5012 \RequirePackage{mikoslides}
5013 </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.

```

5014 \*package>
5015 \bool_if:NT \c__mikoslides_notes_bool {
5016   \RequirePackage{a4wide}
5017   \RequirePackage{marginnote}
5018   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5019   \RequirePackage{mdframed}
5020   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5021   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5022 }
5023 \RequirePackage{stex-compatibility}
5024 \RequirePackage{stex-tikzinput}
5025 \RequirePackage{etoolbox}
5026 \RequirePackage{amssymb}
5027 \RequirePackage{amsmath}
5028 \RequirePackage{comment}
5029 \RequirePackage{textcomp}
5030 \RequirePackage{url}
5031 \RequirePackage{graphicx}
5032 \RequirePackage{pgf}

```

39.2 Notes and Slides

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

```

5033 \bool_if:NT \c__mikoslides_notes_bool {
5034   \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
5035 }

```

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

```

5036 \newcounter{slide}
5037 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5038 \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.

```

5039 \bool_if:NTF \c__mikoslides_notes_bool {
5040   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5041 }{
5042   \excludecomment{note}
5043 }

```

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.

```
5044 \bool_if:NT \c__mikoslides_notes_bool {
5045   \newlength{\slideframewidth}
5046   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5047 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
5048   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5049     \bool_set_true:N #1
5050   }{
5051     \bool_set_false:N #1
5052   }
5053 }
5054 \keys_define:nn{mikoslides / frame}{
5055   label .str_set_x:N = \l__mikoslides_frame_label_str,
5056   allowframebreaks .code:n = {
5057     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
5058   },
5059   allowdisplaybreaks .code:n = {
5060     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
5061   },
5062   fragile .code:n = {
5063     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
5064   },
5065   shrink .code:n = {
5066     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
5067   },
5068   squeeze .code:n = {
5069     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
5070   },
5071   t .code:n = {
5072     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
5073   },
5074 }
5075 \cs_new_protected:Nn \__mikoslides_frame_args:n {
5076   \str_clear:N \l__mikoslides_frame_label_str
5077   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
5078   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
5079   \bool_set_true:N \l__mikoslides_frame_fragile_bool
5080   \bool_set_true:N \l__mikoslides_frame_shrink_bool
5081   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
5082   \bool_set_true:N \l__mikoslides_frame_t_bool
5083   \keys_set:nn { mikoslides / frame }{ #1 }
5084 }
```

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

```
5085 \renewenvironment{frame}[1][]{
5086   \__mikoslides_frame_args:n{#1}
5087   \sffamily
5088   \stepcounter{slide}
5089   \def\@currentlabel{\theslide}
5090   \str_if_empty:NF \l__mikoslides_frame_label_str {
5091     \label{\l__mikoslides_frame_label_str}
```

5092 }
 5093

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

5093 \def\itemize@level{outer}
 5094 \def\itemize@outer{outer}
 5095 \def\itemize@inner{inner}
 5096 \renewcommand\newpage{\addtocounter{framenum}{1}}
 5097 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
 5098 \renewenvironment{itemize}{
 5099 \ifx\itemize@level\itemize@outer
 5100 \def\itemize@label{\$\rhd\$}
 5101 \fi
 5102 \ifx\itemize@level\itemize@inner
 5103 \def\itemize@label{\$\scriptstyle\rhd\$}
 5104 \fi
 5105 \begin{list}
 5106 {\itemize@label}
 5107 {\setlength{\labelsep}{.3em}
 5108 \setlength{\labelwidth}{.5em}
 5109 \setlength{\leftmargin}{1.5em}
 5110 }
 5111 \edef\itemize@level{\itemize@inner}
 5112 }{
 5113 \end{list}
 5114 }

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

5115 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
 5116 }{
 5117 \medskip\miko@slidelabel\end{mdframed}
 5118 }

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

\frametitle

5119 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
 5120 }

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

EdN:21

\pause 21

5121 \bool_if:NT \c__mikoslides_notes_bool {
 5122 \newcommand\pause{
 5123 }

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

nomtext

5124 \bool_if:NTF \c__mikoslides_notes_bool {
 5125 \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
 5126 }{
 5127 \excludecomment{nomtext}
 5128 }

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

```

nomgroup
5129 \bool_if:NTF \c__mikoslides_notes_bool {
5130 \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5131 }{
5132 \excludecomment{nomgroup}
5133 }

ntheorem
5134 \bool_if:NTF \c__mikoslides_notes_bool {
5135 \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5136 }{
5137 \excludecomment{ntheorem}
5138 }

nassertion
5139 \bool_if:NTF \c__mikoslides_notes_bool {
5140 \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5141 }{
5142 \excludecomment{nassertion}
5143 }

nsproof
5144 \bool_if:NTF \c__mikoslides_notes_bool {
5145 \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5146 }{
5147 \excludecomment{nsproof}
5148 }

nexample
5149 \bool_if:NTF \c__mikoslides_notes_bool {
5150 \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5151 }{
5152 \excludecomment{nexample}
5153 }

nparagraph
5154 \bool_if:NTF \c__mikoslides_notes_bool {
5155 \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}{\end{sparagraph}}
5156 }{
5157 \excludecomment{nparagraph}
5158 }

\inputref@*skip We customize the hooks for in \inputref.
5159 \def\inputref@preskip{\smallskip}
5160 \def\inputref@postskip{\medskip}

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

\inputref*
5161 \let\orig@inputref\inputref
5162 \def\inputref{\@ifstar\ninputref\orig@inputref}
5163 \newcommand\ninputref[2] [] {
5164 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5165     \orig@inputref[#1]{#2}
5166   }
5167 }

```

(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>}`.

```

5168 \newlength{\slidelogoheight}
5169
5170 \bool_if:NTF \c_mikoslides_notes_bool {
5171   \setlength{\slidelogoheight}{.4cm}
5172 }{
5173   \setlength{\slidelogoheight}{1cm}
5174 }
5175 \newsavebox{\slidelogo}
5176 \sbox{\slidelogo}{\TeX}
5177 \newrobustcmd{\setslidelogo}[1]{
5178   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5179 }

```

(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.

```

5180 \def\source{Michael Kohlhase}% customize locally
5181 \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.

```

5182 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5183 \newsavebox{\cclogo}
5184 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5185 \newif\ifcchref\cchreffalse
5186 \AtBeginDocument{
5187   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5188 }
5189 \def\licensing{
5190   \ifcchref
5191     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5192   \else
5193     {\usebox{\cclogo}}
5194   \fi
5195 }

```

```

5196 \newrobustcmd{\setlicensing}[2][]{
5197   \def\@url{#1}
5198   \sbox{\cclogo}{\includegraphics[height=\slideologoheight]{#2}}
5199   \ifx\@url\@empty
5200     \def\licensing{\usebox{\cclogo}}
5201   \else
5202     \def\licensing{
5203       \ifcchref
5204         \href{#1}{\usebox{\cclogo}}
5205       \else
5206         {\usebox{\cclogo}}
5207       \fi
5208     }
5209   \fi
5210 }

```

(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

```

5211 \newrobustcmd\miko@slidelabel{
5212   \vbox to \slideologoheight{
5213     \vss\hbox to \slidewidth
5214       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slideologo}}
5215   }
5216 }

```

(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.

```

5217 \def\Gin@mhrepos{}
5218 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5219 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5220 \newrobustcmd\frameimage[2][]{
5221   \stepcounter{slide}
5222   \bool_if:NT \c__mikoslides_frameimages_bool {
5223     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5224     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5225     \begin{center}
5226       \bool_if:NF \c__mikoslides_fiboxed_bool {
5227         \fbox{
5228           \ifx\Gin@ewidth\@empty
5229             \ifx\Gin@mhrepos\@empty
5230               \mhgraphics[width=\slidewidth,#1]{#2}
5231             \else
5232               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5233             \fi
5234           \else% Gin@ewidth empty
5235             \ifx\Gin@mhrepos\@empty
5236               \mhgraphics[#1]{#2}

```

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

```

5237         \else
5238             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5239         \fi
5240     \fi% Gin@ewidth empty
5241 }
5242 }{
5243     \ifx\Gin@ewidth\@empty
5244         \ifx\Gin@mhrepos\@empty
5245             \mhgraphics[width=\slidewidth,#1]{#2}
5246         \else
5247             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5248         \fi
5249         \ifx\Gin@mhrepos\@empty
5250             \mhgraphics[#1]{#2}
5251         \else
5252             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5253         \fi
5254     \fi% Gin@ewidth empty
5255 }
5256 \end{center}
5257 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5258 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5259 }
5260 } % 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.

```

5261 \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

```

5262 \AddToHook{begindocument}{
5263     \definecolor{green}{rgb}{0,.5,0}
5264     \definecolor{purple}{cmyk}{.3,1,0,.17}
5265 }

```

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.

```

5266 % \def\STpresent#1{\textcolor{blue}{#1}}
5267 \def\defemph#1{\textcolor{magenta}{#1}}
5268 \def\symrefemph#1{\textcolor{cyan}{#1}}
5269 \def\compemph#1{\textcolor{blue}{#1}}
5270 \def\titleemph#1{\textcolor{blue}{#1}}
5271 \def\__omtextlec#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.

```

5272 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5273 \def\smalltextwarning{
5274   \pgfuseimage{miko@small@dbend}
5275   \xspace
5276 }
5277 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
5278 \newrobustcmd\textwarning{
5279   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5280   \xspace
5281 }
5282 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5283 \newrobustcmd\bigtextwarning{
5284   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5285   \xspace
5286 }
(End definition for \textwarning. This function is documented on page ??.)
5287 \newrobustcmd\putgraphicsat[3]{
5288   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5289 }
5290 \newrobustcmd\putat[2]{
5291   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5292 }

```

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`.

```

5293 \bool_if:NT \c__mikoslides_sectocframes_bool {
5294   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5295     \newcounter{chapter}\counterwithin*{section}{chapter}
5296   }{
5297     \str_if_eq:VnTF \__mikoslidestopsect{chapter}{
5298       \newcounter{chapter}\counterwithin*{section}{chapter}
5299     }
5300   }
5301 }

```

`\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
5302 \def\part@prefix{}
5303 \@ifpackageloaded{omdoc}{}{
5304   \str_case:VnF \__mikoslidestopsect {
5305     {part}{
5306       \int_set:Nn \l_document_structure_section_level_int {0}
5307       \def\thesection{\arabic{chapter}.\arabic{section}}
5308       \def\part@prefix{\arabic{chapter}.}
5309     }

```

```

5310 {chapter}{
5311   \int_set:Nn \l_document_structure_section_level_int {1}
5312   \def\thesection{\arabic{chapter}.\arabic{section}}
5313   \def\part@prefix{\arabic{chapter}.}
5314 }
5315 }{
5316   \int_set:Nn \l_document_structure_section_level_int {2}
5317   \def\part@prefix{}
5318 }
5319 }
5320
5321 \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

```

5322 \renewenvironment{omgroup}[2][]{
5323   \__document_structure_omgroup_args:n { #1 }
5324   \int_incr:N \l_document_structure_omgroup_level_int
5325   \int_incr:N \l_document_structure_section_level_int
5326   \bool_if:NT \c__mikoslides_sectocframes_bool {
5327     \stepcounter{slide}
5328     \begin{frame}[noframenumbering]
5329     \vfill\Large\centering
5330     \red{
5331       \ifcase\l_document_structure_section_level_int\or
5332         \stepcounter{part}
5333         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5334         \def\currentsectionlevel{\omdoc@part@kw}
5335       \or
5336         \stepcounter{chapter}
5337         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5338         \def\currentsectionlevel{\omdoc@chapter@kw}
5339       \or
5340         \stepcounter{section}
5341         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5342         \def\currentsectionlevel{\omdoc@section@kw}
5343       \or
5344         \stepcounter{subsection}
5345         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5346         \def\currentsectionlevel{\omdoc@subsection@kw}
5347       \or
5348         \stepcounter{subsubsection}
5349         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5350         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5351       \or
5352         \stepcounter{paragraph}
5353         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
5354         \def\currentsectionlevel{\omdoc@paragraph@kw}
5355       \else
5356         \def\__mikoslideslabel{}
5357         \def\currentsectionlevel{\omdoc@paragraph@kw}

```



```

5358         \fi% end ifcase
5359         \_mikoslideslabel%\sref@label@id\_mikoslideslabel
5360         \quad #2%
5361     }%
5362     \vfill%
5363     \end{frame}%
5364 }
5365 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str%
5366 }{}
5367 }

```

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

```

5368 \def\inserttheorembodyfont{\normalfont}
5369 %\bool_if:NF \c__mikoslides_notes_bool {
5370 % \defbeamertemplate{theorem begin}{miko}
5371 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5372 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5373 % \inserttheorem punctuation\inserttheorembodyfont\xspace}
5374 % \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5375 % \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.

```

5376 % \expandafter\def\csname Parent2\endcsname{}
5377 %}
5378
5379 \AddToHook{begindocument}{% this does not work for some reason
5380 \setbeamertemplate{theorems}[ams style]
5381 }
5382 \bool_if:NT \c__mikoslides_notes_bool {
5383 \renewenvironment{columns}[1][1]{%
5384 \par\noindent%
5385 \begin{minipage}%
5386 \slidewidth\centering\leavevmode%
5387 }{%
5388 \end{minipage}\par\noindent%
5389 }%
5390 \newsavebox\columnbox%
5391 \renewenvironment<>{column}[2][1]{%
5392 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5393 }{%
5394 \end{minipage}\end{lrbox}\usebox\columnbox%
5395 }%
5396 }
5397 \bool_if:NTF \c__mikoslides_noproblems_bool {
5398 \newenvironment{problems}{}{}
5399 }{
5400 \excludecomment{problems}
5401 }

```

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.

```

5402 \gdef\printexcursions{}
5403 \newcommand\excursionref[2]{% label, text
5404   \bool_if:NT \c__mikoslides_notes_bool {
5405     \begin{sparagraph}[title=Excursion]
5406       #2 \sref[fallback=the appendix]{#1}.
5407     \end{sparagraph}
5408   }
5409 }
5410 \newcommand\activate@excursion[2][]{
5411   \gappto\printexcursions{\inputref{#1}{#2}}
5412 }
5413 \newcommand\excursion[4][]{% repos, label, path, text
5414   \bool_if:NT \c__mikoslides_notes_bool {
5415     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5416   }
5417 }

```

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

`\excursiongroup`

```

5418 \keys_define:nn{mikoslides / excursiongroup }{
5419   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
5420   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
5421   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5422 }
5423 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5424   \tl_clear:N \l__mikoslides_excursion_intro_tl
5425   \str_clear:N \l__mikoslides_excursion_id_str
5426   \str_clear:N \l__mikoslides_excursion_mhrepos_str
5427   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5428 }
5429 \newcommand\excursiongroup[1][]{
5430   \__mikoslides_excursion_args:n{ #1 }
5431   \ifdefempty\printexcursions{}% only if there are excursions
5432   {\begin{note}
5433     \begin{omgroup}[#1]{Excursions}%
5434     \ifdefempty\l__mikoslides_excursion_intro_tl{\{
5435       \inputref[\l__mikoslides_excursion_mhrepos_str]{
5436         \l__mikoslides_excursion_intro_tl
5437       }
5438     }
5439     \printexcursions%
5440     \end{omgroup}
5441     \end{note}}
5442   }
5443   \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5444   \</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.

```
5445 <*package>
5446 <@@=problems>
5447 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5448 \RequirePackage{l3keys2e,expl-keystr-compat}
5449
5450 \keys_define:nn { problem / pkg }{
5451   notes      .default:n    = { true },
5452   notes      .bool_set:N   = \c__problems_notes_bool,
5453   gnotes     .default:n    = { true },
5454   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5455   hints      .default:n    = { true },
5456   hints      .bool_set:N   = \c__problems_hints_bool,
5457   solutions  .default:n    = { true },
5458   solutions  .bool_set:N   = \c__problems_solutions_bool,
5459   pts        .default:n    = { true },
5460   pts        .bool_set:N   = \c__problems_pts_bool,
5461   min        .default:n    = { true },
5462   min        .bool_set:N   = \c__problems_min_bool,
5463   boxed      .default:n    = { true },
5464   boxed      .bool_set:N   = \c__problems_boxed_bool,
5465   unknown    .code:n       = {}
5466 }
5467 \def\solutionstrue{
5468   \bool_set_true:N \c__problems_solutions_bool
5469 }
5470 \def\solutionsfalse{
5471   \bool_set_false:N \c__problems_solutions_bool
5472 }
5473
5474 \ProcessKeysOptions{ problem / pkg }
```

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

```

5475 \RequirePackage{stex-compatibility}
5476 \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.

```

5477 \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.

```

5478 \def\prob@problem@kw{Problem}
5479 \def\prob@solution@kw{Solution}
5480 \def\prob@hint@kw{Hint}
5481 \def\prob@note@kw{Note}
5482 \def\prob@gnote@kw{Grading}
5483 \def\prob@pt@kw{pt}
5484 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5485 \AddToHook{begindocument}{
5486   \ltx@ifpackageloaded{babel}{
5487     \makeatletter
5488     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5489     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5490       \input{problem-ngerman.ldf}
5491     }
5492     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5493       \input{problem-finnish.ldf}
5494     }
5495     \clist_if_in:NnT \l_tmpa_clist {french}{
5496       \input{problem-french.ldf}
5497     }
5498     \clist_if_in:NnT \l_tmpa_clist {russian}{
5499       \input{problem-russian.ldf}
5500     }
5501     \makeatother
5502   }{}
5503 }

```

40.2 Problems and Solutions

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

```

5504 \keys_define:nn{ problem / problem }{
5505   id      .str_set:x:N = \l__problems_prob_id_str,
5506   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5507   min     .tl_set:N    = \l__problems_prob_min_tl,
5508   title   .tl_set:N    = \l__problems_prob_title_tl,
5509   refnum  .int_set:N    = \l__problems_prob_refnum_int
5510 }
5511 \cs_new_protected:Nn \__problems_prob_args:n {

```

```

5512 \str_clear:N \l__problems_prob_id_str
5513 \tl_clear:N \l__problems_prob_pts_tl
5514 \tl_clear:N \l__problems_prob_min_tl
5515 \tl_clear:N \l__problems_prob_title_tl
5516 \int_zero_new:N \l__problems_prob_refnum_int
5517 \keys_set:nn { problem / problem }{ #1 }
5518 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5519   \let\l__problems_inclprob_refnum_int\undefined
5520 }
5521 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5522 \newcounter{problem}
5523 \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.

```

5524 \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

```

5525 \newcommand\prob@number{
5526   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5527     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5528   }{
5529     \int_if_exist:NTF \l__problems_prob_refnum_int {
5530       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5531     }{
5532       \prob@label\theproblem
5533     }
5534   }
5535 }

```

(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.

```

5536 \newcommand\prob@title[3]{%
5537   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5538     #2 \l__problems_inclprob_title_tl #3
5539   }{
5540     \tl_if_exist:NTF \l__problems_prob_title_tl {
5541       #2 \l__problems_prob_title_tl #3
5542     }{
5543       #1
5544     }
5545   }
5546 }

```

(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.

```
5547 \def\prob@heading{
5548   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5549   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
5550 }
```

(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`

```
5551 \newenvironment{problem}[1][]{
5552   \__problems_prob_args:n{#1}%\sref@target%
5553   \@in@omtexttrue% we are in a statement (for inline definitions)
5554   \stepcounter{problem}\record@problem
5555   \def\current@section@level{\prob@problem@kw}
5556   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5557 }%
5558 {\smallskip}
5559 \bool_if:NT \c__problems_boxed_bool {
5560   \surroundwithmdframed{problem}
5561 }
```

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

```
5562 \def\record@problem{
5563   \protected@write\@auxout{}
5564   {
5565     \string\@problem{\prob@number}
5566     {
5567       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5568         \l__problems_inclprob_pts_tl
5569       }{
5570         \l__problems_prob_pts_tl
5571       }
5572     }%
5573     {
5574       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5575         \l__problems_inclprob_min_tl
5576       }{
5577         \l__problems_prob_min_tl
5578       }
5579     }
5580   }
5581 }
```

(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).

```
5582 \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.

```

5583 \keys_define:nn { problem / solution }{
5584   id          .str_set_x:N = \l__problems_solution_id_str ,
5585   for         .tl_set:N    = \l__problems_solution_for_tl ,
5586   height      .dim_set:N   = \l__problems_solution_height_dim ,
5587   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5588   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5589   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5590 }
5591 \cs_new_protected:Nn \__problems_solution_args:n {
5592   \str_clear:N \l__problems_solution_id_str
5593   \tl_clear:N \l__problems_solution_for_tl
5594   \tl_clear:N \l__problems_solution_srccite_tl
5595   \clist_clear:N \l__problems_solution_creators_clist
5596   \clist_clear:N \l__problems_solution_contributors_clist
5597   \dim_zero:N \l__problems_solution_height_dim
5598   \keys_set:nn { problem / solution }{ #1 }
5599 }

```

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

```

5600 \newcommand\@startsolution[1][ ]{
5601   \__problems_solution_args:n { #1 }
5602   \@in@omtexttrue% we are in a statement.
5603   \bool_if:NF \c__problems_boxed_bool { \hrule }
5604   \smallskip\noindent
5605   {\textbf\prob@solution@kw : \enspace}
5606   \begin{small}
5607   \def\current@section@level{\prob@solution@kw}
5608   \ignorespacesandpars
5609 }

```

\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.

```

5610 \newcommand\startsolutions{
5611   \specialcomment{solution}{\@startsolution}{
5612     \bool_if:NF \c__problems_boxed_bool {
5613       \hrule\medskip
5614     }
5615     \end{small}%
5616   }
5617   \bool_if:NT \c__problems_boxed_bool {
5618     \surroundwithmdframed{solution}
5619   }
5620 }

```

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

\stopsolutions

```

5621 \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.

```

5622 \bool_if:NTF \c__problems_solutions_bool {
5623   \startsolutions
5624 }{
5625   \stopsolutions
5626 }

```

exnote

```

5627 \bool_if:NTF \c__problems_notes_bool {
5628   \newenvironment{exnote}[1][]{
5629     \par\smallskip\hrule\smallskip
5630     \noindent\textbf{\prob@note@kw : }\small
5631   }{
5632     \smallskip\hrule
5633   }
5634 }{
5635   \excludecomment{exnote}
5636 }

```

hint

```

5637 \bool_if:NTF \c__problems_notes_bool {
5638   \newenvironment{hint}[1][]{
5639     \par\smallskip\hrule\smallskip
5640     \noindent\textbf{\prob@hint@kw :~ }\small
5641   }{
5642     \smallskip\hrule
5643   }
5644   \newenvironment{exhint}[1][]{
5645     \par\smallskip\hrule\smallskip
5646     \noindent\textbf{\prob@hint@kw :~ }\small
5647   }{
5648     \smallskip\hrule
5649   }
5650 }{
5651   \excludecomment{hint}
5652   \excludecomment{exhint}
5653 }

```

gnote

```

5654 \bool_if:NTF \c__problems_notes_bool {
5655   \newenvironment{gnote}[1][]{
5656     \par\smallskip\hrule\smallskip
5657     \noindent\textbf{\prob@gnote@kw : }\small
5658   }{
5659     \smallskip\hrule
5660   }
5661 }{
5662   \excludecomment{gnote}
5663 }

```


40.3 Multiple Choice Blocks

```

5664 \newenvironment{mcb}{
5665   \begin{enumerate}
5666 }{
5667   \end{enumerate}
5668 }

```

we define the keys for the mcc macro

```

5669 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5670   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5671     \bool_set_true:N #1
5672   }{
5673     \bool_set_false:N #1
5674   }
5675 }
5676 \keys_define:nn { problem / mcc }{
5677   id          .str_set:N = \l__problems_mcc_id_str ,
5678   feedback    .tl_set:N  = \l__problems_mcc_feedback_tl ,
5679   T           .default:n  = { true } ,
5680   T           .bool_set:N = \l__problems_mcc_t_bool ,
5681   F           .default:n  = { true } ,
5682   F           .bool_set:N = \l__problems_mcc_f_bool ,
5683   Ttext       .code:n     = {
5684     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5685   } ,
5686   Ftext       .code:n     = {
5687     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5688   }
5689 }
5690 \cs_new_protected:Nn \l__problems_mcc_args:n {
5691   \str_clear:N \l__problems_mcc_id_str
5692   \tl_clear:N \l__problems_mcc_feedback_tl
5693   \bool_set_true:N \l__problems_mcc_t_bool
5694   \bool_set_true:N \l__problems_mcc_f_bool
5695   \bool_set_true:N \l__problems_mcc_Ttext_bool
5696   \bool_set_false:N \l__problems_mcc_Ftext_bool
5697   \keys_set:nn { problem / mcc }{ #1 }
5698 }

```

\mcc

```

5699 \newcommand\mcc[2][] {
5700   \l__problems_mcc_args:n{ #1 }
5701   \item #2
5702   \bool_if:NT \c__problems_solutions_bool {
5703     \\\
5704     \bool_if:NT \l__problems_mcc_t_bool {
5705       % TODO!
5706       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
5707     }
5708     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

5709      % TODO!
5710      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5711    }
5712    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5713      !
5714    }{
5715      \l__problems_mcc_feedback_tl
5716    }
5717  }
5718 } %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.

```

5719
5720 \keys_define:nn{ problem / inclproblem }{
5721   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5722   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5723   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5724   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5725   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5726   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5727 }
5728 \cs_new_protected:Nn \__problems_inclprob_args:n {
5729   % \str_clear:N \l__problems_prob_id_str
5730   \tl_clear:N \l__problems_inclprob_pts_tl
5731   \tl_clear:N \l__problems_inclprob_min_tl
5732   \tl_clear:N \l__problems_inclprob_title_tl
5733   \int_zero_new:N \l__problems_inclprob_refnum_int
5734   \str_clear:N \l__problems_inclprob_mhrepos_str
5735   \keys_set:nn { problem / inclproblem }{ #1 }
5736   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5737     \let\l__problems_inclprob_pts_tl\undefined
5738   }
5739   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5740     \let\l__problems_inclprob_min_tl\undefined
5741   }
5742   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5743     \let\l__problems_inclprob_title_tl\undefined
5744   }
5745   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5746     \let\l__problems_inclprob_refnum_int\undefined
5747   }
5748 }
5749
5750 \cs_new_protected:Nn \__problems_inclprob_clear: {
5751   % \str_clear:N \l__problems_prob_id_str
5752   \let\l__problems_inclprob_pts_tl\undefined
5753   \let\l__problems_inclprob_min_tl\undefined

```

```

5754 \let\l__problems_inclprob_title_tl\undefined
5755 \let\l__problems_inclprob_refnum_int\undefined
5756 \let\l__problems_inclprob_mhrepos_str\undefined
5757 }
5758
5759 \newcommand\includeproblem[2][ ]{
5760   \__problems_inclprob_args:n{ #1 }
5761   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5762     \input{#2}
5763   }{
5764     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5765       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5766     }
5767   }
5768   \__problems_inclprob_clear:
5769 }

```

(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.

```

5770 \AddToHook{enddocument}{
5771   \bool_if:NT \c__problems_pts_bool {
5772     \message{Total:~\arabic{pts}~points}
5773   }
5774   \bool_if:NT \c__problems_min_bool {
5775     \message{Total:~\arabic{min}~minutes}
5776   }
5777 }

```

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

```

5778 \def\pts#1{
5779   \bool_if:NT \c__problems_pts_bool {
5780     \marginpar{#1~\prob@pt@kw}
5781   }
5782 }
5783 \def\min#1{
5784   \bool_if:NT \c__problems_min_bool {
5785     \marginpar{#1~\prob@min@kw}
5786   }
5787 }

```

\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.

```

5788 \newcounter{pts}
5789 \def\show@pts{
5790   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5791     \bool_if:NT \c__problems_pts_bool {
5792       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5793       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5794     }
5795   }{
5796     \tl_if_exist:NT \l__problems_prob_pts_tl {
5797       \bool_if:NT \c__problems_pts_bool {
5798         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5799         \addtocounter{pts}{\l__problems_prob_pts_tl}
5800       }
5801     }
5802   }
5803 }

```

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

\show@min

```

5804 \newcounter{min}
5805 \def\show@min{
5806   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5807     \bool_if:NT \c__problems_min_bool {
5808       \marginpar{\l__problems_inclprob_pts_tl;min}
5809       \addtocounter{min}{\l__problems_inclprob_min_tl}
5810     }
5811   }{
5812     \tl_if_exist:NT \l__problems_prob_min_tl {
5813       \bool_if:NT \c__problems_min_bool {
5814         \marginpar{\l__problems_prob_min_tl;min}
5815         \addtocounter{min}{\l__problems_prob_min_tl}
5816       }
5817     }
5818   }
5819 }
5820 \</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.

```
5821 <@@=hwexam>
5822 <*cls>
5823 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5824 \RequirePackage{l3keys2e,expl-keystr-compatible}
5825 \DeclareOption*{
5826   \PassOptionsToClass{\CurrentOption}{omdoc}
5827   \PassOptionsToPackage{\CurrentOption}{stex}
5828   \PassOptionsToPackage{\CurrentOption}{hwexam}
5829   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5830 }
5831 \ProcessOptions
```

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

```
5832 \LoadClass{omdoc}
5833 \RequirePackage{stex}
5834 \RequirePackage{hwexam}
5835 \RequirePackage{tikzinput}
5836 \RequirePackage{graphicx}
5837 \RequirePackage{a4wide}
5838 \RequirePackage{amssymb}
5839 \RequirePackage{amstext}
5840 \RequirePackage{amsmath}
```

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

```

5841 \newcommand\assig@default@type{\hwexam@assignment@kw}
5842 \def\document@hwexamtype{\assig@default@type}
5843 <@@=document_structure>
5844 \keys_define:nn { document-structure / document }{
5845 id .str_set_x:N = \c_document_structure_document_id_str,
5846 hwexamtype .tl_set:N = \document@hwexamtype
5847 }
5848 <@@=hwexam>
5849 </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.

```
5850 \*package>
5851 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5852 \RequirePackage{l3keys2e,expl-keystr-compat}
5853
5854 \newif\iftest\testfalse
5855 \DeclareOption{test}{\testtrue}
5856 \newif\ifmultiple\multiplefalse
5857 \DeclareOption{multiple}{\multipletrue}
5858 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5859 \ProcessOptions
```

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

```
5860 \RequirePackage{keyval}[1997/11/10]
5861 \RequirePackage{problem}
```

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

```
5862 \newcommand\hwexam@assignment@kw{Assignment}
5863 \newcommand\hwexam@given@kw{Given}
5864 \newcommand\hwexam@due@kw{Due}
5865 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5866 blank~for~extra~space}%
5867 \newcommand\correction@probs@kw{prob.}%
5868 \newcommand\correction@pts@kw{total}%
5869 \newcommand\correction@reached@kw{reached}%
5870 \newcommand\correction@sum@kw{Sum}%
5871 \newcommand\correction@grade@kw{grade}%
5872 \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

```

5873 \AddToHook{begindocument}{
5874 \ltx@ifpackageloaded{babel}{
5875 \makeatletter
5876 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5877 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5878 \input{hwexam-ngerman.ldf}
5879 }
5880 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5881 \input{hwexam-finnish.ldf}
5882 }
5883 \clist_if_in:NnT \l_tmpa_clist {french}{
5884 \input{hwexam-french.ldf}
5885 }
5886 \clist_if_in:NnT \l_tmpa_clist {russian}{
5887 \input{hwexam-russian.ldf}
5888 }
5889 \makeatother
5890 }{}
5891 }
5892

```

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.

```

5893 \newcounter{assignment}
5894 \numberproblemsin{assignment}
5895 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5896 \keys_define:nn { hwexam / assignment } {
5897 id .str_set:N = \l__hwexam_assign_id_str,
5898 number .int_set:N = \l__hwexam_assign_number_int,
5899 title .tl_set:N = \l__hwexam_assign_title_tl,
5900 type .tl_set:N = \l__hwexam_assign_type_tl,
5901 given .tl_set:N = \l__hwexam_assign_given_tl,
5902 due .tl_set:N = \l__hwexam_assign_due_tl,
5903 loadmodules .code:n = {
5904 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5905 }
5906 }
5907 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5908 \str_clear:N \l__hwexam_assign_id_str
5909 \int_set:Nn \l__hwexam_assign_number_int {-1}
5910 \tl_clear:N \l__hwexam_assign_title_tl
5911 \tl_clear:N \l__hwexam_assign_type_tl
5912 \tl_clear:N \l__hwexam_assign_given_tl
5913 \tl_clear:N \l__hwexam_assign_due_tl
5914 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```



```

5915 \keys_set:nn { hwexam / assignment }{ #1 }
5916 }

```

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.

```

5917 \newcommand\given@due[2]{
5918 \bool_lazy_all:nF {
5919 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5920 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5921 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5922 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5923 }{ #1 }
5924
5925 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5926 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5927 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5928 }
5929 }{
5930 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5931 }
5932
5933 \bool_lazy_or:nnF {
5934 \bool_lazy_and_p:nn {
5935 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5936 }{
5937 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5938 }
5939 }{
5940 \bool_lazy_and_p:nn {
5941 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5942 }{
5943 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5944 }
5945 }{ ,~ }
5946
5947 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5948 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5949 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5950 }
5951 }{
5952 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5953 }
5954
5955 \bool_lazy_all:nF {
5956 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5957 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5958 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5959 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5960 }{ #2 }
5961 }

```

`\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.

```

5962 \newcommand\assignment@title[3]{
5963 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5964 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5965 #1
5966 }{
5967 #2\l__hwexam_assign_title_tl#3
5968 }
5969 }{
5970 #2\l__hwexam_inclassassign_title_tl#3
5971 }
5972 }

```

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

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

```

5973 \newcommand\assignment@number{
5974 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5975 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5976 \int_use:N \l__hwexam_assign_number_int
5977 }
5978 }{
5979 \int_use:N \l__hwexam_inclassassign_number_int
5980 }
5981 }

```

(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.

```

5982 \newenvironment{assignment}[1][]{
5983 \__hwexam_assignment_args:n { #1 }
5984 %\sref@target
5985 \let\__hwexamnum\l__hwexam_assign_number_int
5986 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5987 \stepcounter{assignment}
5988 }{
5989 \setcounter{assignment}{\int_use:N\__hwexamnum}
5990 }
5991 \setcounter{problem}{0}
5992 \def\current@section@level{\document@hwexamtype}
5993 %\sref@label@id{\document@hwexamtype \thesection}
5994 \begin{@assignment}
5995 }{
5996 \end{@assignment}
5997 }

```

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

```

5998 \def\__hwexasstitle{
5999 \protect\document@hwexamtype~\arabic{assignment}
6000 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6001 }
6002 \ifmultiple
6003 \newenvironment{@assignment}{
6004 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6005 \begin{omgroup}[loadmodules]{\__hwexasstitle}
6006 }{
6007 \begin{omgroup}{\__hwexasstitle}
6008 }
6009 }{
6010 \end{omgroup}
6011 }

```

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

```

6012 \else
6013 \newenvironment{@assignment}{
6014 \begin{center}\bf
6015 \Large@title\strut\
6016 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}
6017 \large\given@due{--;\}\{;\}
6018 \end{center}
6019 }{}
6020 \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.

```

6021 \keys_define:nn { hwexam / inclassignment } {
6022 %id .str_set_x:N = \l__hwexam_assign_id_str,
6023 number .int_set:N = \l__hwexam_inclassign_number_int,
6024 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6025 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6026 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6027 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6028 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6029 }
6030 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6031 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6032 \tl_clear:N \l__hwexam_inclassign_title_tl
6033 \tl_clear:N \l__hwexam_inclassign_type_tl
6034 \tl_clear:N \l__hwexam_inclassign_given_tl
6035 \tl_clear:N \l__hwexam_inclassign_due_tl
6036 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6037 \keys_set:nn { hwexam / inclassignment }{ #1 }
6038 }
6039 \__hwexam_inclassignment_args:n {}
6040
6041 \newcommand\inputassignment[2][{}]{

```

```

6042 \_hwexam_inclassassignment_args:n { #1 }
6043 \str_if_empty:NTF \l__hwexam_inclasssign_mhrepos_str {
6044 \input{#2}
6045 }{
6046 \stex_in_repository:nn{\l__hwexam_inclasssign_mhrepos_str}{
6047 \input{\mhp{path}{\l__hwexam_inclasssign_mhrepos_str}{#2}}
6048 }
6049 }
6050 \_hwexam_inclassassignment_args:n {}
6051 }
6052 \newcommand\includeassignment[2][]{
6053 \newpage
6054 \inputassignment[#1]{#2}
6055 }

```

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

42.4 Typesetting Exams

\quizheading

```

6056 \ExplSyntaxOff
6057 \newcommand\quizheading[1]{%
6058 \def\@tas{#1}%
6059 \large\noindent NAME: \hspace{8cm} MAILBOX:\|[2ex]%
6060 \ifx\@tas\@empty\else%
6061 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\|[2ex]%
6062 \fi%
6063 }
6064 \ExplSyntaxOn

```

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

\testheading

```

6065 \keys_define:nn { hwexam / testheading } {
6066 min .tl_set:N = \l__hwexam_testheading_min_tl,
6067 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
6068 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
6069 }
6070 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6071 \tl_clear:N \l__hwexam_testheading_min_tl
6072 \tl_clear:N \l__hwexam_testheading_duration_tl
6073 \tl_clear:N \l__hwexam_testheading_reqpts_tl
6074 \keys_set:nn { hwexam / testheading }{ #1 }
6075 }
6076 \newenvironment{testheading}[1][]{
6077 \_hwexam_testheading_args:n{ #1 }
6078 \noindent\large{Name:~\hfill
6079 Matriculation Number:\hspace*{2cm}\strut}\|[1ex]
6080 \begin{center}
6081 \Large\textbf{\@title}\|[1ex]
6082 \large\@date\|[3ex]
6083 \end{center}
6084 \textbf{You~have~

```

```

6085 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
6086 {\l__hwexam_testheading_min_tl}~minutes
6087 }{
6088 {\l__hwexam_testheading_duration_tl}
6089 }~
6090 (sharp)~for~the~test
6091 };\\
6092 Write~the~solutions~to~the~sheet.
6093 \par\noindent
6094 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
6095 \advance\check@time by -\theassignment@totalmin
6096 The~estimated~time~for~solving~this~exam~is~
6097 {\theassignment@totalmin}~minutes,~
6098 leaving~you~{\the\check@time}~minutes~for~revising~
6099 your~exam.
6100
6101 \par\noindent
6102 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
6103 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
6104 You~can~reach~{\theassignment@totalpts}~points~if~you~
6105 solve~all~problems.~You~will~only~need~
6106 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
6107 i.e.\ {\the\bonus@pts}~points~are~bonus~points.
6108 \vfill
6109 \begin{center}
6110 {
6111 \Large\em You~have~ample~time,~so~take~it~slow~
6112 and~avoid~rushing~to~mistakes!\\[2ex]
6113 Different~problems~test~different~skills~and~
6114 knowledge,~so~do~not~get~stuck~on~one~problem.
6115 }
6116 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\\[3ex]
6117 \end{center}
6118 }{
6119 \newpage
6120 }

```

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

\testspace

```

6121 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6122 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6123 \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.

```

6124 <@@=problems>
6125 \renewcommand\@problem[3]{
6126 \stepcounter{assignment@probs}
6127 \def\__problemspts{#2}
6128 \ifx\__problemspts\@empty\else
6129 \addtocounter{assignment@totalpts}{#2}
6130 \fi
6131 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6132 \xdef\correction@probs{\correction@probs & #1}%
6133 \xdef\correction@pts{\correction@pts & #2}
6134 \xdef\correction@reached{\correction@reached &}
6135 }
6136 <@@=hwexam>

```

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

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

```

6137 \newcounter{assignment@probs}
6138 \newcounter{assignment@totalpts}
6139 \newcounter{assignment@totalmin}
6140 \def\correction@probs{\correction@probs@kw}%
6141 \def\correction@pts{\correction@pts@kw}%
6142 \def\correction@reached{\correction@reached@kw}%
6143 \def\after@correction@table{}%
6144 \stepcounter{assignment@probs}
6145 \newcommand\correction@table{
6146 \resizebox{\textwidth}{!}{%
6147 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6148 &\multicolumn{\theassignment@probs}{c|}|%|
6149 {\footnotesize\correction@forgrading@kw} &\\ \hline
6150 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6151 \correction@pts & \theassignment@totalpts & \\ \hline
6152 \correction@reached & & \[.7cm]\hline
6153 \end{tabular}}
6154 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6155 </package>

```

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

42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}

```

```
\newcommand\uhr{{\uhrfont\char65}}  
\newcommand\warnschild{{\warnschildfont\char 65}}  
\newcommand\hardA{\warnschild}  
\newcommand\longA{\uhr}  
\newcommand\thinkA{\denker}  
\newcommand\discussA{\biertglas}
```