

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	95
28.1	The module environment	98
28.2	Invoking modules	104
29	STeX-Module Inheritance Implementation	106
29.1	SMS Mode	106
29.2	Inheritance	110
30	STeX-Symbols Implementation	115
30.1	Symbol Declarations	115
30.2	Notations	122
31	STeX-Terms Implementation	131
31.1	Symbol Invocations	131
31.2	Terms	134
31.3	Notation Components	140
32	STeX-Structural Features Implementation	143
32.1	Imports with modification	143
32.2	The feature environment	150
32.3	Features	152
33	STeX-Statements Implementation	157
33.1	Definitions	157
33.2	Assertions	160
33.3	Examples	162
33.4	Logical Paragraphs	164

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

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

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

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using Semantic Macros

TODO

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

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

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smglob/calculus
narration-base: http://mathhub.info/smglob/calculus
dependencies: smglom/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}{\comp1}

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> *	
<code>\stex_path_if_absolute:NTF</code> *	

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 .sms-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⟩}`

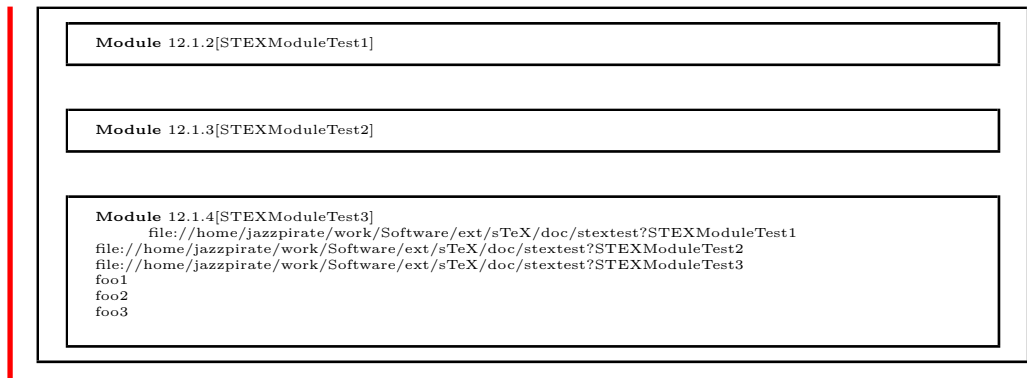
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\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

<u><code>\symdecl</code></u>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</code>
------------------------------	--

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

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by TeX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
 - b a *variable* argument. Is treated by TeX like an i-argument, but an application is turned into an OMBind in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

`\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}_\comp
```

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  $\mult{a,\plus{\frac{ab}{ac}}}$ 
 $\displaystyle \plus{a,\mult{b,c}}$  and  $\mult{a,\plus{\frac{ab}{ac}}}$ 
\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

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

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

18.2.5 Proof End Markers

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

$$\backslash\text{proofend}$$

$$\backslash\text{sProofEndSymbol}$$

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

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

18.2.6 Configuration of the Presentation

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

Environment	configuration macro	value
<code>sproof</code>	<code>\spf@proof@kw</code>	Proof
<code>sketchproof</code>	<code>\spf@sketchproof@kw</code>	ProofSketch

Figure 1: Configuration Hooks for Semantic Proof Markup

$$\backslash\text{pstlabelstyle}$$

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

This package supplies an infrastructure for writing OMDOC documents in $\text{\texttt{LaTeX}}$. This includes a simple structure sharing mechanism for $\text{\texttt{sTeX}}$ 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 $\text{\texttt{sTeX}}$ sources, or after translation.

21.1 Introduction

$\text{\texttt{sTeX}}$ is a version of $\text{\texttt{TeX/LaTeX}}$ that allows to markup $\text{\texttt{TeX/LaTeX}}$ documents semantically without leaving the document format, essentially turning $\text{\texttt{TeX/LaTeX}}$ 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 $\text{\texttt{sTeX}}$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\text{\texttt{sTeX}}$ 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 L^AT_EX_ML transformation.

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

```

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

```

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup`

⁹EDNOTE: integrate with latexml's XMRef in the Math mode.
²We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

The `\skipomgroup` “skips an `omgroup`”, i.e. it just steps the respective sectioning counter. This macro is useful, when we want to keep two documents in sync structurally, so that section numbers match up: Any section that is left out in one becomes a `\skipomgroup`.

`\currentsectionlevel`

`\CurrentSectionLevel`

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

21.2.3 Ignoring Inputs

`ignore`
`showignores`

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

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

is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

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

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

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

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [LMH].

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

- | | |
|---------------------------|---|
| <code>slides</code> | • 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{<label>}` for that.

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

23.2 The User Interface

23.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [<code>Kohlhase:mss</code>].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [<code>Kohlhase:metakeys</code>]
	for details and customization options).

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

23.2.2 Problems and Solutions

problem The main environment provided by the **problem** package is (surprise surprise) the **problem** environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys **id** as an identifier that can be reference later, **pts** for the points to be gained from this exercise in homework or quiz situations, **min** for the estimated minutes needed to solve the problem, and finally **title** for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

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

Example 5: A marked up Problem

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

```
Problem0.0 ()
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


Solution: Four, two in the front seats, and two in the back.


---


```

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

Also, sometimes, we want content (e.g. in an exam with master solutions) conditional on whether solutions are shown. This can be done with the `\ifsolutions` conditional.

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

The `\includeproblem` macro can be used to include a problem from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one problem in the include file). The keys `title`, `min`, and `pts` specify the problem title, the estimated minutes for solving the problem and the points to be gained, and their values (if given) overwrite the ones specified in the `problem` environment in the included file.

23.2.5 Reporting Metadata

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

The `\min` and `\pts` macros allow to specify (i.e. to print to the margin) the distribution of time and reward to parts of a problem, if the `pts` and `pts` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

23.3 Limitations

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

1. none reported yet

```

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

```

Problem0.0 ()

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

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

Problem0.0 ()

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV
Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

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

25.3 Messages and logging

```

43 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

25.4 Persistence

78 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

25.5 HTML Annotations

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

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

```

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

```

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

```

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

```



```

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

```

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

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

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_ML, R_US_TE_X, p_DF_LA_TE_X).

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

```

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

```

```

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

```

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

25.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

301 </package>

```

Chapter 26

STEX -MathHub Implementation

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

26.1 Generic Path Handling

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

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

```

328     }
329     \seq_set_eq:NN #1 \l_tmpa_tl
330   }
331   \stex_path_canonicalize:N #1
332 }
333 }
334 \cs_generate_variant:Nn \stex_path_from_string:Nn
335 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
336 \cs_new_protected:Nn \stex_path_to_string:NN {
337   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
338 }
339
340 \cs_new:Nn \stex_path_to_string:N {
341   \seq_use:Nn #1 /
342 }

```

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

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
343 \str_const:Nn \c__stex_path_dot_str {.}
344 \str_const:Nn \c__stex_path_up_str {...}

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

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

```

```

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

```

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

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

```

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

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

26.3 File Hooks and Tracking

407 `<@@=stex_files>`

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack` keeps track of file changes

408 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

410 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

411 `\c_stex_mainfile_str`

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

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

412 `\seq_gclear_new:N\g_stex_currentfile_seq`

413 `\AddToHook{file/before}{`

414 `\stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}`

415 `\stex_path_if_absolute:NTF\g_stex_currentfile_seq{`

416 `\exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}`

417 `}{`

418 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

419 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

420 `}`

421 `}`

422 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

423 `\exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq`

424 `}`

425 `\AddToHook{file/after}{`

426 `\seq_if_empty:NF\g__stex_files_stack{`

427 `\seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq`

428 `}`

429 `\seq_if_empty:NTF\g__stex_files_stack{`

430 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

431 `}{`

432 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

433 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

434 `}`

435 `}`

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

26.4 MathHub Repositories

```

436 <@@=stex_mathhub>

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

```

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

```

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

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
473 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

509 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

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

```

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

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

```

```

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

```

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

```

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

```

```

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

```

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

`\mhpath`

```

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

```

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

`\libinput`

```

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

```

```

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

```

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

```

721 </package>

```


Chapter 27

STEX -References Implementation

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

27.1 Document URIs and URLs

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

```

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

```

```

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

```

27.2 Setting Reference Targets

```

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

```

```

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

```

27.3 Using References

```

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

```

```

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

```

Chapter 28

STEX -Modules Implementation

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

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

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

`\stex_add_constant_to_current_module:n`

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

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

`\stex_collect_imports:n`

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

```

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

```

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

`\stex_add_import_to_current_module:n`

```

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

```

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

`\stex_modules_compute_namespace:nN`

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

```

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

```

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

Stores its return values in:


```
\l_stex_modules_ns_str
\l_stex_modules_subpath_str
```

```
1054 \str_new:N \l_stex_modules_ns_str
1055 \str_new:N \l_stex_modules_subpath_str
```

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

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

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

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

28.1 The module environment

module arguments:

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

```

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

```

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

```

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

```

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

Are we in a nested module?

```

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

```

Next, we determine the language of the module:

```

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

```

```

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

```

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

```

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

```

We load the metatheory:

```

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

```

```

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

```

(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}

1210 \int_new:N \l_stex_module_group_depth_int
1211 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1212   \stex_reactivate_macro:N \STEXexport
1213   \stex_reactivate_macro:N \importmodule
1214   \stex_reactivate_macro:N \symdecl
1215   \stex_reactivate_macro:N \notation
1216   \stex_reactivate_macro:N \symdef
1217   \stex_module_setup:nn{#1}{#2}
1218 }
1219 \stex_debug:nn{modules}{
1220   New~module:\\
1221   Namespace:~\l_stex_module_ns_str\\
1222   Name:~\l_stex_module_name_str\\
1223   Language:~\l_stex_module_lang_str\\
1224   Signature:~\l_stex_module_sig_str\\
1225   Metatheory:~\l_stex_module_meta_str\\
1226   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1227 }
1228
1229 \seq_put_right:Nx \l_stex_all_modules_seq {
1230   \l_stex_module_ns_str ? \l_stex_module_name_str
1231 }
1232
1233 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1234 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1235
1236
1237 \stex_if_smsmode:TF {

```

```

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

```

1256 \cs_new_protected:Nn \_stex_modules_end_module: {
1257 % \str_set:Nx \l_tmpa_str {
1258 % c_stex_module_
1259 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1260 % \prop_item:Nn \l_stex_current_module_prop { name }
1261 % _prop
1262 % }
1263 %^^A \prop_new:c { \l_tmpa_str }
1264 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1265 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module_
1266 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1267 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1268 \NewDocumentEnvironment { @module } { 0 } { m } {
1269 \par
1270 \_stex_modules_begin_module:nn{#1}{#2}
1271 } {
1272 \_stex_modules_end_module:
1273 \stex_if_smsmode:TF {
1274 % \exp_args:Nx \stex_add_to_sms:n {
1275 % \prop_gset_from_keyval:cn {
1276 % c_stex_module_
1277 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1278 % \prop_item:Nn \l_stex_current_module_prop { name }
1279 % _prop
1280 % } {
1281 % name = \prop_item:cn { \l_tmpa_str } { name } ,
1282 % ns = \prop_item:cn { \l_tmpa_str } { ns } ,
1283 % file = \prop_item:cn { \l_tmpa_str } { file } ,

```

```

1284 %      lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1285 %      sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1286 %      meta      = \prop_item:cn { \l_tmpa_str } { meta }
1287 %    }
1288 %  }
1289 }{
1290   \end{stex_annotate_env}
1291 }
1292 }

```

\stex_modules_heading: Code for document headers

```

1293 \cs_if_exist:NTF \thesection {
1294   \newcounter{module}[section]
1295 }{
1296   \newcounter{module}
1297 }
1298
1299 \bool_if:NT \c_stex_showmods_bool {
1300   \latexml_if:F { \RequirePackage{mdframed} }
1301 }
1302
1303 \cs_new_protected:Nn \stex_modules_heading: {
1304   \stepcounter{module}
1305   \par
1306   \bool_if:NT \c_stex_showmods_bool {
1307     \noindent{\textbf{Module} ~
1308       \cs_if_exist:NT \thesection {\thesection.}
1309       \themodule ~ [\l_stex_module_name_str]
1310     }
1311     \str_if_empty:NTF \l_stex_module_title_str {
1312       }{
1313         \quad(\l_stex_module_title_str)\hfill
1314       }\par
1315     }
1316     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1317     % TODO
1318     \stex_ref_new_doc_target:n \l_stex_module_name_str
1319   }

```

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

Finally:

```

1320 \NewDocumentEnvironment { module } { 0 } { m } {
1321   \bool_if:NT \c_stex_showmods_bool {
1322     \begin{mdframed}
1323   }
1324   \begin{@module}[#1]{#2}
1325     \stex_modules_heading:
1326   }{
1327     \end{@module}
1328     \bool_if:NT \c_stex_showmods_bool {
1329       \end{mdframed}
1330     }
1331   }

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1332 \NewDocumentCommand \STEXModule { m } {
1333   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1334   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1335   \tl_set:Nn \l_tmpa_tl {
1336     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1337   }
1338   \seq_map_inline:Nn \l_stex_all_modules_seq {
1339     \str_set:Nn \l_tmpb_str { ##1 }
1340     \str_if_eq:eeT { \l_tmpa_str } {
1341       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1342     } {
1343       \seq_map_break:n {
1344         \tl_set:Nn \l_tmpa_tl {
1345           \stex_invoke_module:n { ##1 }
1346         }
1347       }
1348     }
1349   }
1350   \l_tmpa_tl
1351 }
1352
1353 \cs_new_protected:Nn \stex_invoke_module:n {
1354   \stex_debug:nn{modules}{Invoking~module~#1}
1355   \peek_charcode_remove:NTF ! {
1356     \__stex_modules_invoke_uri:nN { #1 }
1357   } {
1358     \peek_charcode_remove:NTF ? {
1359       \__stex_modules_invoke_symbol:nn { #1 }
1360     } {
1361       \msg_error:nnx{stex}{error/syntax}{
1362         ?~or~!~expected~after~
1363         \c_backslash_str STEXModule{#1}
1364       }
1365     }
1366   }
1367 }
1368
1369 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1370   \str_set:Nn #2 { #1 }
1371 }
1372
1373 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1374   \stex_invoke_symbol:n{#1?#2}
1375 }

```

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

```

\stex_activate_module:n
1376 \bool_new:N \l_stex_in_meta_bool
1377 \bool_set_false:N \l_stex_in_meta_bool

```

```

1378 \cs_new_protected:Nn \stex_activate_module:n {
1379   \stex_debug:nn{modules}{Activating~module~#1}
1380   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1381     \msg_error:nnn{stex}{error/concllictingmodules}{ #1 }
1382   }
1383   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1384     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1385     \use:c{ c_stex_module_#1_code }
1386   }
1387 }

```

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

```

1388 </package>

```


Chapter 29

sTeX -Module Inheritance Implementation

```
1389 <*package>
1390
1391 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1392
```

29.1 SMS Mode

```
1393 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1394 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1395 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1396 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1397
1398 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1399   \makeatletter
1400   \makeatother
1401   \ExplSyntaxOn
1402   \ExplSyntaxOff
1403 }
1404
1405 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1406   \symdef
1407   \importmodule
1408   \notation
1409   \symdecl
1410   \STEXexport
1411 }
1412
1413 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1414   \tl_to_str:n {
1415     module,
1416     @module
```

```

1417 }
1418 }

```

(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

```

```

1419 \bool_new:N \g__stex_smsmode_bool
1420 \bool_set_false:N \g__stex_smsmode_bool
1421 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1422   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1423 }

```

(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

```

```

1424 \bool_new:N \g__stex_smsmode_catcode_bool
1425 \bool_set_false:N \g__stex_smsmode_catcode_bool
1426 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1427   \bool_if:NTF \g__stex_smsmode_catcode_bool
1428   \prg_return_true: \prg_return_false:
1429 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1430 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1431   \stex_if_smsmode:T {
1432     \__stex_smsmode_if_catcodes:F {
1433       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1434       \exp_after:wN \char_gset_active_eq:NN
1435       \c_backslash_str \__stex_smsmode_cs:
1436       \tex_global:D \char_set_catcode_active:N \
1437       \tex_global:D \char_set_catcode_other:N $
1438       \tex_global:D \char_set_catcode_other:N ^
1439       \tex_global:D \char_set_catcode_other:N _
1440       \tex_global:D \char_set_catcode_other:N &
1441       \tex_global:D \char_set_catcode_other:N ##
1442     }
1443   }
1444 } \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.

```

1445 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1446   \__stex_smsmode_if_catcodes:T {
1447     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1448     \exp_after:wN \tex_global:D \exp_after:wN
1449     \char_set_catcode_escape:N \c_backslash_str
1450     \tex_global:D \char_set_catcode_math_toggle:N $
1451     \tex_global:D \char_set_catcode_math_superscript:N ^
1452     \tex_global:D \char_set_catcode_math_subscript:N _
1453     \tex_global:D \char_set_catcode_alignment:N &
1454     \tex_global:D \char_set_catcode_parameter:N ##
1455   }
1456 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1457 \cs_new_protected:Nn \stex_in_smsmode:nn {
1458   \vbox_set:Nn \l_tmpa_box {
1459     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1460     \bool_gset_true:N \g__stex_smsmode_bool
1461     \stex_smsmode_set_codes:
1462     #2
1463     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1464     \stex_if_smsmode:F {
1465       \__stex_smsmode_unset_codes:
1466     }
1467   }
1468   \box_clear:N \l_tmpa_box
1469 }

```

(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:

```

1470 \cs_new_protected:Nn \_stex_smsmode_cs: {
1471   \str_clear:N \l_tmpa_str
1472   \peek_analysis_map_inline:n {
1473     % #1: token (one expansion)
1474     % #2: charcode
1475     % #3 catcode
1476     \token_if_eq_charcode:NNTF ##3 B {
1477       % token is a letter
1478       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1479     } {
1480       \str_if_empty:NTF \l_tmpa_str {
1481         % we don't allow (or need) single non-letter CSs
1482         % for now
1483         \peek_analysis_map_break:
1484       }{
1485         \str_if_eq:onTF \l_tmpa_str { begin } {
1486           \peek_analysis_map_break:n {
1487             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1488           }
1489         } {
1490           \str_if_eq:onTF \l_tmpa_str { end } {
1491             \peek_analysis_map_break:n {
1492               \exp_after:wN \_stex_smsmode_checkend:n ##1
1493             }
1494           } {
1495             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1496             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1497               \g_stex_smsmode_allowedmacros_tl
1498               { \use:c{\l_tmpa_str} } {
1499               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1500               \peek_analysis_map_break:n {
1501                 \exp_after:wN \l_tmpa_tl ##1
1502               }

```

```

1503     } {
1504         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1505         \g_stex_smsmode_allowedmacros_escape_tl
1506         { \use:c{\l_tmpa_str} } {
1507             \__stex_smsmode_unset_codes:
1508             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1509             % TODO \__stex_smsmode_rescan_cs:
1510             \int_compare:nNnTF {##2} = {92} {
1511                 \peek_analysis_map_break:n {
1512                     \__stex_smsmode_unset_codes:
1513                     \__stex_smsmode_rescan_cs:
1514                 }
1515             } {
1516                 \peek_analysis_map_break:n {
1517                     \exp_after:wN \l_tmpa_tl ##1
1518                 }
1519             }
1520         } {
1521             \int_compare:nNnTF {##2} = {92} {
1522                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1523             } {
1524                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1525             }
1526         }
1527     }
1528 }
1529 }
1530 }
1531 }
1532 }
1533 }

```

(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:

```

1534 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1535     \str_clear:N \l_tmpb_str
1536     \peek_analysis_map_inline:n {
1537         \token_if_eq_charcode:NNTF ##3 B {
1538             % token is a letter
1539             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1540         } {
1541             \peek_analysis_map_break:n {
1542                 \exp_after:wN \use:c \exp_after:wN {
1543                     \exp_after:wN \l_tmpa_str\exp_after:wN
1544                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1545             }
1546         }
1547     }
1548 }

```

(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.

```

1549 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1550   \str_set:Nn \l_tmpa_str { #1 }
1551   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1552     \__stex_smsmode_unset_codes:
1553     \begin{#1}
1554   }
1555 }
```

(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.

```

1556 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1557   \str_set:Nn \l_tmpa_str { #1 }
1558   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1559     \end{#1}
1560   }
1561 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1562 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1563 \cs_new_protected:Nn \stex_import_module_uri:nn {
1564   \str_set:Nx \l_stex_import_archive_str { #1 }
1565   \str_set:Nn \l_stex_import_path_str { #2 }
1566
1567   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1568   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1569   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1570
1571   \stex_modules_current_namespace:
1572   \bool_lazy_all:nTF {
1573     {\str_if_empty_p:N \l_stex_import_archive_str}
1574     {\str_if_empty_p:N \l_stex_import_path_str}
1575     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1576   }{
1577     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1578     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1579   }{
1580     \str_if_empty:NT \l_stex_import_archive_str {
1581       \prop_if_exist:NT \l_stex_current_repository_prop {
1582         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1583       }
1584     }
1585     \str_if_empty:NTF \l_stex_import_archive_str {
1586       \str_if_empty:NF \l_stex_import_path_str {
1587         \str_set:Nx \l_stex_import_ns_str {
1588           \l_stex_module_ns_str / \l_stex_import_path_str
1589         }
1590       }
1591     }
```

```

1591   }{
1592     \stex_require_repository:n \l_stex_import_archive_str
1593     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1594     \l_stex_import_ns_str
1595     \str_if_empty:NF \l_stex_import_path_str {
1596       \str_set:Nx \l_stex_import_ns_str {
1597         \l_stex_import_ns_str / \l_stex_import_path_str
1598       }
1599     }
1600   }
1601 }
1602 }

```

(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 1603 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1604 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1605 \str_new:N \l_stex_import_path_str
                           1606 \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>}
1607 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1608   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1609
1610     % archive
1611     \str_set:Nx \l_tmpa_str { #2 }
1612     \str_if_empty:NTF \l_tmpa_str {
1613       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1614     } {
1615       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1616       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1617       \seq_put_right:Nn \l_tmpa_seq { source }
1618     }
1619
1620     % path
1621     \str_set:Nx \l_tmpb_str { #3 }
1622     \str_if_empty:NTF \l_tmpb_str {
1623       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1624
1625       \ltx@ifpackageloaded{babel} {
1626         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1627           { \language } \l_tmpb_str {
1628           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1629         }
1630       } {
1631         \str_clear:N \l_tmpb_str
1632       }
1633
1634       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1635       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1636         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1637     }{
1638       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1639       \IfFileExists{ \l_tmpa_str.tex }{
1640         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1641       }{
1642         % try english as default
1643         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1644         \IfFileExists{ \l_tmpa_str.en.tex }{
1645           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1646         }{
1647           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1648         }
1649       }
1650     }
1651
1652   } {
1653     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1654     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1655
1656     \ltx@ifpackageloaded{babel} {
1657       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1658         { \language } \l_tmpb_str {
1659         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1660       }
1661     } {
1662       \str_clear:N \l_tmpb_str
1663     }
1664
1665     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1666
1667     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1668     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1669       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1670     }{
1671       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1672       \IfFileExists{ \l_tmpa_str/#4.tex }{
1673         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1674       }{
1675         % try english as default
1676         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1677         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1678           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1679         }{
1680           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1681           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1682             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1683           }{
1684             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1685             \IfFileExists{ \l_tmpa_str.tex }{
1686               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1687             }{
1688               % try english as default
1689               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1690               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1691         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1692     }{
1693         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1694     }
1695 }
1696 }
1697 }
1698 }
1699 }
1700 }
1701
1702 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1703     \seq_clear:N \l_stex_all_modules_seq
1704     \str_clear:N \l_stex_current_module_str
1705     \str_set:Nx \l_tmpb_str { #2 }
1706     \str_if_empty:NF \l_tmpb_str {
1707         \stex_set_current_repository:n { #2 }
1708     }
1709     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1710     \input { \g__stex_importmodule_file_str }
1711 }
1712
1713 \stex_if_module_exists:nF { #1 ? #4 } {
1714     \msg_error:nnx{stex}{error/unknownmodule}{
1715         #1?#4~(in~file~\g__stex_importmodule_file_str)
1716     }
1717 }
1718 }
1719 \stex_activate_module:n { #1 ? #4 }
1720 }

```

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

`\importmodule`

```

1721 \NewDocumentCommand \importmodule { 0{} m } {
1722     \stex_import_module_uri:nn { #1 } { #2 }
1723     \stex_debug:nn{modules}{Importing~module:~
1724         \l_stex_import_ns_str ? \l_stex_import_name_str
1725     }
1726     \stex_if_smsmode:F {
1727         \stex_import_require_module:nnnn
1728         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1729         { \l_stex_import_path_str } { \l_stex_import_name_str }
1730         \stex_annotate_invisible:nnn
1731         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1732     }
1733     \exp_args:Nx \stex_add_to_current_module:n {
1734         \stex_import_require_module:nnnn
1735         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1736         { \l_stex_import_path_str } { \l_stex_import_name_str }
1737     }
1738     \exp_args:Nx \stex_add_import_to_current_module:n {
1739         \l_stex_import_ns_str ? \l_stex_import_name_str
1740     }

```



```

1741 \stex_smsmode_set_codes:
1742 }
1743 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1744 \NewDocumentCommand \usemodule { 0{} m } {
1745   \stex_if_smsmode:F {
1746     \stex_import_module_uri:nn { #1 } { #2 }
1747     \stex_import_require_module:nnnn
1748     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1749     { \l_stex_import_path_str } { \l_stex_import_name_str }
1750     \stex_annotate_invisible:nnn
1751     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1752   }
1753   \stex_smsmode_set_codes:
1754 }

```

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

```

1755 \endpackage

```

Chapter 30

STEX -Symbols Implementation

```
1756 <*package>
1757
1758 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1759
```

Warnings and error messages

```
1760
```

30.1 Symbol Declarations

```
1761 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1762 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1768 \NewDocumentCommand \STEXsymbol { m } {
1769   \stex_get_symbol:n { #1 }
1770   \exp_args:No
1771   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1772 }
1773
```

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

symdecl arguments:

```
1768 \keys_define:nn { stex / symdecl } {
1769   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1770   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1771   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1772   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1773   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1774   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1775   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1776   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1777 }
```

```

1778
1779 \bool_new:N \l_stex_symdecl_make_macro_bool
1780
1781 \cs_new_protected:Nn \__stex_symdecl_args:n {
1782   \str_clear:N \l_stex_symdecl_name_str
1783   \str_clear:N \l_stex_symdecl_args_str
1784   \bool_set_false:N \l_stex_symdecl_local_bool
1785   \tl_clear:N \l_stex_symdecl_type_tl
1786   \tl_clear:N \l_stex_symdecl_definiens_tl
1787
1788   \keys_set:nn { stex / symdecl } { #1 }
1789 }

```

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

```

1790
1791 \NewDocumentCommand \symdecl { s O{} m } {
1792   \__stex_symdecl_args:n { #2 }
1793   \IfBooleanTF #1 {
1794     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1795   } {
1796     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1797   }
1798   \stex_symdecl_do:n { #3 }
1799   \stex_smsmode_set_codes:
1800 }
1801 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1802 \cs_new_protected:Nn \stex_symdecl_do:n {
1803   \stex_if_in_module:F {
1804     % TODO throw error? some default namespace?
1805   }
1806
1807   \str_if_empty:NT \l_stex_symdecl_name_str {
1808     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1809   }
1810
1811   \prop_if_exist:cT { l_stex_symdecl_
1812     \l_stex_current_module_str ?
1813     \l_stex_symdecl_name_str
1814     _prop
1815   }{
1816     % TODO throw error (beware of circular dependencies)
1817   }
1818
1819   \prop_clear:N \l_tmpa_prop
1820   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1821   \seq_clear:N \l_tmpa_seq
1822   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1823   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1824

```

```

1825 \exp_args:No \stex_add_constant_to_current_module:n {
1826   \l_stex_symdecl_name_str
1827 }
1828
1829 % arity/args
1830 \int_zero:N \l_tmpb_int
1831
1832 \bool_set_true:N \l_tmpa_bool
1833 \str_map_inline:Nn \l_stex_symdecl_args_str {
1834   \token_case_meaning:NnF ##1 {
1835     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1836     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1837     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1838     {\tl_to_str:n a} {
1839       \bool_set_false:N \l_tmpa_bool
1840       \int_incr:N \l_tmpb_int
1841     }
1842     {\tl_to_str:n B} {
1843       \bool_set_false:N \l_tmpa_bool
1844       \int_incr:N \l_tmpb_int
1845     }
1846   }{
1847     \msg_set:nnn{stex}{error/wrongargs}{
1848       args~value~in~symbol~declaration~for~
1849       \l_stex_current_module_str ?
1850       \l_stex_symdecl_name_str ~
1851       needs~to~be~
1852       i,~a,~b~or~B,~but~##1~given
1853     }
1854     \msg_error:nn{stex}{error/wrongargs}
1855   }
1856 }
1857 \bool_if:NTF \l_tmpa_bool {
1858   % possibly numeric
1859   \str_if_empty:NTF \l_stex_symdecl_args_str {
1860     \prop_put:Nnn \l_tmpa_prop { args } {}
1861     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1862   }{
1863     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1864     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1865     \str_clear:N \l_tmpa_str
1866     \int_step_inline:nn \l_tmpa_int {
1867       \str_put_right:Nn \l_tmpa_str i
1868     }
1869     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1870   }
1871 } {
1872   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1873   \prop_put:Nnx \l_tmpa_prop { arity }
1874     { \str_count:N \l_stex_symdecl_args_str }
1875 }
1876 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1877
1878

```

```

1879 % semantic macro
1880
1881 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1882   \exp_args:Nx \stex_do_aftergroup:n {
1883     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1884       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1885     }}
1886   }
1887
1888   \bool_if:NF \l_stex_symdecl_local_bool {
1889     \exp_args:Nx \stex_add_to_current_module:n {
1890       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1891         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1892       } }
1893     }
1894   }
1895 }
1896
1897 % add to all symbols
1898
1899 \bool_if:NF \l_stex_symdecl_local_bool {
1900   \exp_args:Nx \stex_add_to_current_module:n {
1901     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1902       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1903     }
1904   }
1905 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1906 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1907 %   }
1908 }
1909
1910 \stex_debug:nn{symbols}{New~symbol:~
1911   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1912   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1913   Args:~\prop_item:Nn \l_tmpa_prop { args }
1914 }
1915
1916 % circular dependencies require this:
1917
1918 \prop_if_exist:cF {
1919   l_stex_symdecl_
1920   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1921   _prop
1922 } {
1923   \prop_set_eq:cN {
1924     l_stex_symdecl_
1925     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1926     _prop
1927   } \l_tmpa_prop
1928 }
1929
1930 \seq_clear:c {
1931   l_stex_symdecl_
1932   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

1933   _notations
1934 }
1935
1936 \bool_if:NF \l_stex_symdecl_local_bool {
1937   \exp_args:Nx
1938   \stex_add_to_current_module:n {
1939     \seq_clear:c {
1940       l_stex_symdecl_
1941       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1942       _notations
1943     }
1944     \prop_set_from_keyval:cn {
1945       l_stex_symdecl_
1946       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1947       _prop
1948     } {
1949       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1950       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1951       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1952       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1953       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1954       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1955     }
1956   }
1957 }
1958
1959 \stex_if_smsmode:TF {
1960   \bool_if:NF \l_stex_symdecl_local_bool {
1961     % \exp_args:Nx \stex_add_to_sms:n {
1962     %   \prop_set_from_keyval:cn {
1963     %     l_stex_symdecl_
1964     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1965     %     _prop
1966     %   } {
1967     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1968     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1969     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1970     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1971     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1972     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1973     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1974     %   }
1975     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1976     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1977     %   }
1978     % }
1979   }
1980 }{
1981   \exp_args:Nx \stex_do_aftergroup:n {
1982     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1983       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1984     }
1985   }
1986   \stex_if_do_html:T {

```

```

1987 \stex_annotate_invisible:nnn {symdecl} {
1988   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1989 } {
1990   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
1991   \stex_annotate_invisible:nnn{args}{}{
1992     \prop_item:Nn \l_tmpa_prop { args }
1993   }
1994   \stex_annotate_invisible:nnn{macroname}{#1}{}
1995   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1996     \stex_annotate_invisible:nnn{definiens}{}
1997     { $\l_stex_symdecl_definiens_tl$ }
1998   }
1999 }
2000 }
2001 }
2002 }

```

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

`\stex_get_symbol:n`

```

2003 \str_new:N \l_stex_get_symbol_uri_str
2004
2005 \cs_new_protected:Nn \stex_get_symbol:n {
2006   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2007     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2008   }{
2009     % argument is a string
2010     % is it a command name?
2011     \cs_if_exist:cTF { #1 }{
2012       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2013       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2014       \str_if_empty:NNTF \l_tmpa_str {
2015         \exp_args:Nx \cs_if_eq:NNTF {
2016           \tl_head:N \l_tmpa_tl
2017         } \stex_invoke_symbol:n {
2018           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2019         }{
2020           \__stex_symdecl_get_symbol_from_string:n { #1 }
2021         }
2022       } {
2023         \__stex_symdecl_get_symbol_from_string:n { #1 }
2024       }
2025     }{
2026       % argument is not a command name
2027       \__stex_symdecl_get_symbol_from_string:n { #1 }
2028       % \l_stex_all_symbols_seq
2029     }
2030   }
2031 }
2032
2033 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2034   \str_set:Nn \l_tmpa_str { #1 }
2035   \bool_set_false:N \l_tmpa_bool
2036   \stex_if_in_module:T {

```

```

2037 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2038 \bool_set_true:N \l_tmpa_bool
2039 \str_set:Nx \l_stex_get_symbol_uri_str {
2040 \l_stex_current_module_str ? #1
2041 }
2042 }
2043 }
2044 \bool_if:NF \l_tmpa_bool {
2045 \tl_set:Nn \l_tmpa_tl {
2046 \msg_set:nnn{stex}{error/unknownsymbol}{
2047 No~symbol~#1~found!
2048 }
2049 \msg_error:nn{stex}{error/unknownsymbol}
2050 }
2051 \str_set:Nn \l_tmpa_str { #1 }
2052 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2053 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2054 \str_set:Nn \l_tmpb_str { ##1 }
2055 \str_if_eq:eeT { \l_tmpa_str } {
2056 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2057 } {
2058 \seq_map_break:n {
2059 \tl_set:Nn \l_tmpa_tl {
2060 \str_set:Nn \l_stex_get_symbol_uri_str {
2061 ##1
2062 }
2063 }
2064 }
2065 }
2066 }
2067 \l_tmpa_tl
2068 }
2069 }
2070
2071 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2072 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2073 { \tl_tail:N \l_tmpa_tl }
2074 \tl_if_single:NTF \l_tmpa_tl {
2075 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2076 \exp_after:wN \str_set:Nn \exp_after:wN
2077 \l_stex_get_symbol_uri_str \l_tmpa_tl
2078 }{
2079 % TODO
2080 % tail is not a single group
2081 }
2082 }{
2083 % TODO
2084 % tail is not a single group
2085 }
2086 }

```

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

30.2 Notations

```

2087 <@@=stex_notation>

notation arguments:
2088 \keys_define:nn { stex / notation } {
2089   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2090   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2091   prec .str_set_x:N = \l__stex_notation_prec_str ,
2092   op .tl_set:N = \l__stex_notation_op_tl ,
2093   primary .bool_set:N = \l__stex_notation_primary_bool ,
2094   primary .default:n = {true} ,
2095   unknown .code:n = \str_set:Nx
2096     \l__stex_notation_variant_str \l_keys_key_str
2097 }
2098
2099 \cs_new_protected:Nn \stex_notation_args:n {
2100   \str_clear:N \l__stex_notation_lang_str
2101   \str_clear:N \l__stex_notation_variant_str
2102   \str_clear:N \l__stex_notation_prec_str
2103   \tl_clear:N \l__stex_notation_op_tl
2104   \bool_set_false:N \l__stex_notation_primary_bool
2105
2106   \keys_set:nn { stex / notation } { #1 }
2107 }

```

\notation

```

2108 \NewDocumentCommand \notation { 0{ } m } {
2109   \stex_notation_args:n { #1 }
2110   \tl_clear:N \l_stex_symdecl_definiens_tl
2111   \stex_get_symbol:n { #2 }
2112   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2113 }
2114 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2115 \cs_new_protected:Nn \stex_notation_do:nn {
2116   \let\l_stex_current_symbol_str\relax
2117   \prop_set_eq:Nc \l_tmpa_prop {
2118     \l_stex_symdecl_ #1 _prop
2119   }
2120
2121   \prop_clear:N \l_tmpb_prop
2122   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2123   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2124   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2125
2126   % precedences
2127   \seq_clear:N \l_tmpb_seq
2128   \exp_args:NNno
2129   \str_if_empty:NTF \l__stex_notation_prec_str {
2130     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2131     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2132     \exp_args:NNnx
2133     \prop_put:Nno \l_tmpb_prop { opprec }
2134     { \neginfprec }
2135   }{
2136     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2137   }
2138 } {
2139   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2140     \exp_args:NNnx
2141     \prop_put:Nno \l_tmpb_prop { opprec }
2142     { \neginfprec }
2143     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2144     \int_step_inline:nn { \l_tmpa_str } {
2145       \exp_args:NNx
2146       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2147     }
2148   }{
2149     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2150     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2151       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2152       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2153         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2154         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2155         \seq_map_inline:Nn \l_tmpa_seq {
2156           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2157         }
2158       }
2159       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2160     }{
2161       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2162       \int_compare:nNnTF \l_tmpa_str = 0 {
2163         \exp_args:NNnx
2164         \prop_put:Nno \l_tmpb_prop { opprec }
2165         { \infprec }
2166       }{
2167         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2168       }
2169     }
2170   }
2171 }
2172
2173 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2174 \int_step_inline:nn { \l_tmpa_str } {
2175   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2176     \exp_args:NNx
2177     \seq_put_right:Nn \l_tmpb_seq {
2178       \prop_item:Nn \l_tmpb_prop { opprec }
2179     }
2180   }
2181 }
2182
2183 \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2184 \tl_clear:N \l_tmpa_tl
2185

```

```

2186 \int_compare:nNnTF \l_tmpa_str = 0 {
2187   \exp_args:NNe
2188   \cs_set:Npn \l__stex_notation_macrocode_cs {
2189     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2190     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2191     { \prop_item:Nn \l_tmpb_prop { opprec } }
2192     { \exp_not:n { #2 } }
2193   }
2194   \__stex_notation_final:
2195 }{
2196   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2197   \str_if_in:NnTF \l_tmpb_str b {
2198     \exp_args:Nne \use:nn
2199     {
2200       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2201       \cs_set:Npn \l_tmpa_str } { {
2202         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2203         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2204         { \prop_item:Nn \l_tmpb_prop { opprec } }
2205         { \exp_not:n { #2 } }
2206       } }
2207   }{
2208     \str_if_in:NnTF \l_tmpb_str B {
2209       \exp_args:Nne \use:nn
2210       {
2211         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2212         \cs_set:Npn \l_tmpa_str } { {
2213           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2214           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2215           { \prop_item:Nn \l_tmpb_prop { opprec } }
2216           { \exp_not:n { #2 } }
2217         } }
2218     }{
2219       \exp_args:Nne \use:nn
2220       {
2221         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2222         \cs_set:Npn \l_tmpa_str } { {
2223           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2224           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2225           { \prop_item:Nn \l_tmpb_prop { opprec } }
2226           { \exp_not:n { #2 } }
2227         } }
2228     }
2229   }
2230
2231   \int_zero:N \l_tmpa_int
2232   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2233   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2234   \__stex_notation_arguments:
2235 }
2236 }

```

(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

```

2237 \cs_new_protected:Nn \_stex_notation_arguments: {
2238   \int_incr:N \l_tmpa_int
2239   \str_if_empty:NTF \l_tmpa_str {
2240     \_stex_notation_final:
2241   }{
2242     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2243     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2244     \str_if_eq:VnTF \l_tmpb_str a {
2245       \_stex_notation_argument_assoc:n
2246     }{
2247       \str_if_eq:VnTF \l_tmpb_str B {
2248         \_stex_notation_argument_assoc:n
2249       }{
2250         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2251         \tl_put_right:Nx \l_tmpa_tl {
2252           { \_stex_term_math_arg:nnn
2253             { \int_use:N \l_tmpa_int }
2254             { \l_tmpb_str }
2255             { ####\int_use:N \l_tmpa_int }
2256           }
2257         }
2258         \_stex_notation_arguments:
2259       }
2260     }
2261   }
2262 }
```

(End definition for `_stex_notation_arguments:.`)

`_stex_notation_argument_assoc:n`

```

2263 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2264   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2265   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2266   \tl_put_right:Nx \l_tmpa_tl {
2267     { \_stex_term_math_assoc_arg:nnnn
2268       { \int_use:N \l_tmpa_int }
2269       { \l_tmpb_str }
2270       \exp_args:No \exp_not:n
2271       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2272       { ####\int_use:N \l_tmpa_int }
2273     }
2274   }
2275   \_stex_notation_arguments:
2276 }
```

(End definition for `_stex_notation_argument_assoc:n.`)

`_stex_notation_final:` Called after processing all notation arguments

```

2277 \cs_new_protected:Nn \_stex_notation_final: {
2278   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2279   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2280   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2281   \exp_args:Nne \use:nn
```

```

2282 {
2283 \cs_generate_from_arg_count:cNnn {
2284   stex_notation_ \l_tmpa_str \c_hash_str
2285   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2286   _cs
2287 }
2288 \cs_set:Npn \l_tmpb_str } { {
2289   \exp_after:wN \exp_after:wN \exp_after:wN
2290   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2291   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2292 } }
2293
2294 \tl_if_empty:NF \l__stex_notation_op_tl {
2295   \cs_set:cpx {
2296     stex_op_notation_ \l_tmpa_str \c_hash_str
2297     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2298     _cs
2299   } {
2300     \_stex_term_oms:nnn {
2301       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2302       \l__stex_notation_lang_str
2303     }{
2304       \l_tmpa_str
2305     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2306   }
2307 }
2308
2309 \exp_args:Ne
2310 \stex_add_to_current_module:n {
2311   \cs_generate_from_arg_count:cNnn {
2312     stex_notation_ \l_tmpa_str \c_hash_str
2313     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2314     _cs
2315   } \cs_set:Npn {\l_tmpb_str} {
2316     \exp_after:wN \exp_after:wN \exp_after:wN
2317     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2318     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2319   }
2320   \tl_if_empty:NF \l__stex_notation_op_tl {
2321     \cs_set:cpn {
2322       stex_op_notation_ \l_tmpa_str \c_hash_str
2323       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2324       _cs
2325     } {
2326       \_stex_term_oms:nnn {
2327         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2328         \l__stex_notation_lang_str
2329       }{
2330         \l_tmpa_str
2331       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2332     }
2333   }
2334 }
2335

```

```

2336 \seq_put_right:cx {
2337   l_stex_symdecl_
2338   \prop_item:Nn \l_tmpb_prop { symbol }
2339   _notations
2340 } {
2341   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2342 }
2343
2344 \stex_debug:nn{symbols}{
2345   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2346   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2347   Operator~precedence:~
2348   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2349   Argument~precedences:~
2350   \seq_use:Nn \l_tmpa_seq {,~}^^J
2351   Notation: \cs_meaning:c {
2352     stex_notation_ \l_tmpa_str \c_hash_str
2353     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2354     _cs
2355   }
2356 }
2357
2358 \prop_set_eq:cN {
2359   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2360   \c_hash_str \l__stex_notation_lang_str _prop
2361 } \l_tmpb_prop
2362
2363 \exp_args:Ne
2364 \stex_add_to_current_module:n {
2365   \seq_put_right:cn {
2366     l_stex_symdecl_
2367     \prop_item:Nn \l_tmpb_prop { symbol }
2368     _notations
2369   } {
2370     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2371   }
2372   \prop_set_from_keyval:cn {
2373     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2374     \c_hash_str \l__stex_notation_lang_str _prop
2375   } {
2376     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2377     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2378     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2379     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2380     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2381   }
2382 }
2383
2384 \stex_if_smsmode:TF {
2385   \stex_smsmode_set_codes:
2386   % \exp_args:Nx \stex_add_to_sms:n {
2387   %   \prop_set_from_keyval:cn {
2388   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2389   %     \c_hash_str \l__stex_notation_lang_str _prop

```

```

2390 %      } {
2391 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2392 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2393 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2394 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2395 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2396 %      }
2397 %  }
2398 }{
2399
2400 % HTML annotations
2401 \stex_if_do_html:T {
2402     \stex_annotate_invisible:nnn { notation }
2403     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2404         \stex_annotate_invisible:nnn { notationfragment }
2405         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2406         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2407         \stex_annotate_invisible:nnn { precedence }
2408         { \prop_item:Nn \l_tmpb_prop { opprec };
2409           \seq_use:Nn \l_tmpa_seq { x }
2410         }{}
2411
2412         \int_zero:N \l_tmpa_int
2413         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2414         \tl_clear:N \l_tmpa_tl
2415         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2416             \int_incr:N \l_tmpa_int
2417             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2418             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2419             \str_if_eq:VnTF \l_tmpb_str a {
2420                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2421                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2422                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2423                 } }
2424             }{
2425                 \str_if_eq:VnTF \l_tmpb_str B {
2426                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2427                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2428                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2429                     } }
2430                 }{
2431                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2432                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2433                     } }
2434                 }
2435             }
2436         }
2437         \stex_annotate_invisible:nnn { notationcomp }{}{
2438             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2439             $ \exp_args:Nno \use:nn { \use:c {
2440                 stex_notation_ \l_stex_current_symbol_str
2441                 \c_hash_str \l__stex_notation_variant_str
2442                 \c_hash_str \l__stex_notation_lang_str _cs
2443             } } { \l_tmpa_tl } $

```

```

2444     }
2445   }
2446 }
2447 }
2448 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2449 \keys_define:nn { stex / setnotation } {
2450   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2451   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2452   unknown .code:n = \str_set:Nx
2453     \l__stex_notation_variant_str \l_keys_key_str
2454 }
2455
2456 \cs_new_protected:Nn \_stex_setnotation_args:n {
2457   \str_clear:N \l__stex_notation_lang_str
2458   \str_clear:N \l__stex_notation_variant_str
2459   \keys_set:nn { stex / setnotation } { #1 }
2460 }
2461
2462 \NewDocumentCommand \setnotation {m m} {
2463   \stex_get_symbol:n { #1 }
2464   \_stex_setnotation_args:n { #2 }
2465   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2466     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2467     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2468       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2469     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2470       { \c_hash_str }
2471     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2472       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2473     \exp_args:Nx \stex_add_to_current_module:n {
2474       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2475         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2476     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2477       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2478     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2479       { \c_hash_str }
2480   }
2481   \stex_debug:nn {notations}{
2482     Setting~default~notation~
2483     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2484     \l_stex_get_symbol_uri_str \
2485     \expandafter\meaning\csname
2486     l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2487   }
2488   }{
2489     % todo throw error
2490   }
2491 }
2492

```

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

\symdef

```
2493 \keys_define:nn { stex / symdef } {
2494   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2495   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2496   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2497   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2498   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2499   op        .tl_set:N    = \l__stex_notation_op_tl ,
2500   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2501   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2502   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2503   unknown   .code:n      = \str_set:Nx
2504             \l__stex_notation_variant_str \l_keys_key_str
2505 }
2506
2507 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2508   \str_clear:N \l_stex_symdecl_name_str
2509   \str_clear:N \l_stex_symdecl_args_str
2510   \bool_set_false:N \l_stex_symdecl_local_bool
2511   \tl_clear:N \l_stex_symdecl_type_tl
2512   \tl_clear:N \l_stex_symdecl_definiens_tl
2513   \str_clear:N \l__stex_notation_lang_str
2514   \str_clear:N \l__stex_notation_variant_str
2515   \str_clear:N \l__stex_notation_prec_str
2516   \tl_clear:N \l__stex_notation_op_tl
2517
2518   \keys_set:nn { stex / symdef } { #1 }
2519 }
2520
2521 \NewDocumentCommand \symdef { 0{} m } {
2522   \__stex_notation_symdef_args:n { #1 }
2523   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2524   \stex_symdecl_do:n { #2 }
2525   \exp_args:Nx \stex_notation_do:nn {
2526     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2527   }
2528 }
2529 \stex_deactivate_macro:Nn \symdef {module~environments}
2530
2531 (End definition for \symdef. This function is documented on page 37.)
2532 \endpackage
```

Chapter 31

STEX -Terms Implementation

```
2531 <*package>
2532
2533 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2534
2535 <@@=stex_terms>
2536
2537 Warnings and error messages
2538 \msg_new:nnn{stex}{error/nonotation}{
2539   Symbol~#1~invoked,~but~has~no~notation~#2!
2540 }
2541 \msg_new:nnn{stex}{error/notationarg}{
2542   Error~in~parsing~notation~#1
2543 }
2544 \msg_new:nnn{stex}{error/noop}{
2545   Symbol~#1~has~no~operator~notation~for~notation~#2
2546 }
```

31.1 Symbol Invocations

Arguments:

```
2546 \keys_define:nn { stex / terms } {
2547   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2548   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2549   unknown .code:n = \str_set:Nx
2550     \l__stex_terms_variant_str \l_keys_key_str
2551 }
2552
2553 \cs_new_protected:Nn \__stex_terms_args:n {
2554   \str_clear:N \l__stex_terms_lang_str
2555   \str_clear:N \l__stex_terms_variant_str
2556   \str_clear:N \l__stex_terms_prec_str
2557   \tl_clear:N \l__stex_terms_op_tl
2558 }
2559 \keys_set:nn { stex / terms } { #1 }
```

2560 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2561 \cs_new_protected:Nn \stex_invoke_symbol:n {
2562   \if_mode_math:
2563     \exp_after:wN \__stex_terms_invoke_math:n
2564   \else:
2565     \exp_after:wN \__stex_terms_invoke_text:n
2566   \fi: { #1 }
2567 }
```

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

__stex_terms_invoke_math:n

```
2568 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2569   \peek_charcode_remove:NTF ! {
2570     \peek_charcode:NTF [ {
2571       \__stex_terms_invoke_op:nw { #1 }
2572     }{
2573       \peek_charcode_remove:NTF ! {
2574         \peek_charcode:NTF [ {
2575           \__stex_terms_invoke_op_custom:nw
2576         }{
2577           % TODO throw error
2578         }
2579       }{
2580         \__stex_terms_invoke_op:nw { #1 } []
2581       }
2582     }
2583   }{
2584     \peek_charcode_remove:NTF * {
2585       \__stex_terms_invoke_text:n { #1 }
2586     }{
2587       \peek_charcode:NTF [ {
2588         \__stex_terms_invoke_math:nw { #1 }
2589       }{
2590         \__stex_terms_invoke_math:nw { #1 } []
2591       }
2592     }
2593   }
2594 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2595 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2596   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2597     \stex_highlight_term:nn{#1}{#2}
2598   }
2599 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

__stex_terms_invoke_op:nw

```

2600 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2601   \__stex_terms_args:n { #2 }
2602   \cs_if_exist:cTF {
2603     stex_op_notation_ #1 \c_hash_str
2604     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2605   }{
2606     \csname stex_op_notation_ #1 \c_hash_str
2607       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2608     \endcsname
2609   }{
2610     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2611   }
2612 }

```

(End definition for __stex_terms_invoke_op:nw.)

__stex_terms_invoke_math:nw

```

2613 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2614   \__stex_terms_args:n { #2 }
2615   \seq_if_empty:cTF {
2616     l_stex_symdecl_ #1 _notations
2617   } {
2618     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2619   } {
2620     \seq_if_in:cxTF {
2621       l_stex_symdecl_ #1 _notations
2622     }
2623     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2624       \str_set:Nn \l_stex_current_symbol_str { #1 }
2625       \use:c{
2626         stex_notation_ #1 \c_hash_str
2627         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2628         _cs
2629       }
2630     }{
2631       \str_if_empty:NTF \l__stex_terms_variant_str {
2632         \str_if_empty:NTF \l__stex_terms_lang_str {
2633           \seq_get_left:cN {
2634             l_stex_symdecl_ #1 _notations
2635           } \l_tmpa_str
2636           \str_set:Nn \l_stex_current_symbol_str { #1 }
2637           \use:c{
2638             stex_notation_ #1 \c_hash_str \l_tmpa_str
2639             _cs
2640           }
2641         }{
2642           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2643             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2644           }
2645         }
2646       }{
2647         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2648           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2649     }
2650   }
2651 }
2652 }
2653 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2654 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2655   \peek_charcode_remove:NTF ! {
2656     \stex_term_custom:nn { #1 } { }
2657   }{
2658     \prop_set_eq:Nc \l_tmpa_prop {
2659       l_stex_symdecl_ #1 _prop
2660     }
2661     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2662     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2663   }
2664 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2665 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2666 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2667 \int_new:N \l__stex_terms_downprec
2668 \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
2669 \tl_set:Nn \l__stex_terms_left_bracket_str (
2670 \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

```

2671 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2672   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2673     \bool_set_false:N \l__stex_terms_brackets_done_bool
2674     #2
2675   } {
2676     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2677       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2678         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2679         \dobrackets { #2 }
2680       }

```

```

2681     }{ #2 }
2682   }
2683 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2684 \bool_new:N \l__stex_terms_brackets_done_bool
2685 %\RequirePackage{scalerel}
2686 \cs_new_protected:Npn \dobrackets #1 {
2687   %\ThisStyle{\if D\m@switch
2688   %   \exp_args:Nnx \use:nn
2689   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2690   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2691   % \else
2692   \exp_args:Nnx \use:nn
2693   {
2694     \bool_set_true:N \l__stex_terms_brackets_done_bool
2695     \int_set:Nn \l__stex_terms_downprec \infprec
2696     \l__stex_terms_left_bracket_str
2697     #1
2698   }
2699   {
2700     \bool_set_false:N \l__stex_terms_brackets_done_bool
2701     \l__stex_terms_right_bracket_str
2702     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2703   }
2704   %\fi}
2705 }

```

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

`\withbrackets`

```

2706 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2707   \exp_args:Nnx \use:nn
2708   {
2709     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2710     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2711     #3
2712   }
2713   {
2714     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2715     {\l__stex_terms_left_bracket_str}
2716     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2717     {\l__stex_terms_right_bracket_str}
2718   }
2719 }

```

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

`\STEXinvisible`

```

2720 \cs_new_protected:Npn \STEXinvisible #1 {
2721   \stex_annotate_invisible:n { #1 }
2722 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2723 \cs_new_protected:Nn \_stex_term_oms:nnn {
2724   \stex_annotate:nnn{ OMID }{ #2 }{
2725     \stex_highlight_term:nn { #1 } { #3 }
2726   }
2727 }
2728
2729 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2730   \__stex_terms_maybe_brackets:nn { #3 }{
2731     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2732   }
2733 }
```

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

`_stex_term_math_oma:nnnn`

```

2734 \cs_new_protected:Nn \_stex_term_oma:nnn {
2735   \stex_annotate:nnn{ OMA }{ #2 }{
2736     \stex_highlight_term:nn { #1 } { #3 }
2737   }
2738 }
2739
2740 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2741   \__stex_terms_maybe_brackets:nn { #3 }{
2742     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2743   }
2744 }
```

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

`_stex_term_math_omb:nnnn`

```

2745 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2746   \stex_annotate:nnn{ OMBIND }{ #2 }{
2747     \stex_highlight_term:nn { #1 } { #3 }
2748   }
2749 }
2750
2751 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2752   \__stex_terms_maybe_brackets:nn { #3 }{
2753     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2754   }
2755 }
```

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

`_stex_term_math_arg:nnn`

```

2756 \cs_new_protected:Nn \_stex_term_arg:nn {
2757   \stex_unhighlight_term:n {
2758     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2759   }
2760 }
```

```

2761 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2762   \exp_args:Nnx \use:nn
2763     { \int_set:Nn \l__stex_terms_downprec { #2 }
2764       \stex_term_arg:nn { #1 }{ #3 }
2765     }
2766     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2767   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2768 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2769   \clist_set:Nn \l_tmpa_clist{ #4 }
2770   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2771     \tl_set:Nn \l_tmpa_tl { #4 }
2772   }{
2773     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2774     \clist_reverse:N \l_tmpa_clist
2775     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2776
2777     \clist_map_inline:Nn \l_tmpa_clist {
2778       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2779         \exp_args:Nno
2780           \l_tmpa_cs { ##1 } \l_tmpa_tl
2781       }
2782     }
2783
2784   }
2785   \exp_args:Nnno
2786   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2787 }

```

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

`\stex_term_custom:nn`

```

2788 \cs_new_protected:Nn \stex_term_custom:nn {
2789   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2790   \str_set:Nn \l_tmpa_str { #2 }
2791   \tl_clear:N \l_tmpa_tl
2792   \int_zero:N \l_tmpa_int
2793   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2794   \__stex_terms_custom_loop:
2795 }

```

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

`__stex_terms_custom_loop:`

```

2796 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2797   \bool_set_false:N \l_tmpa_bool
2798   \bool_while_do:nn {
2799     \str_if_eq_p:ee X {
2800       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2801     }
2802   }{
2803     \int_incr:N \l_tmpa_int

```



```

2804 }
2805
2806 \peek_charcode:NTF [ {
2807   % notation/text component
2808   \__stex_terms_custom_component:w
2809 } {
2810   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2811     % all arguments read => finish
2812     \__stex_terms_custom_final:
2813   } {
2814     % arguments missing
2815     \peek_charcode_remove:NTF * {
2816       % invisible, specific argument position or both
2817       \peek_charcode:NTF [ {
2818         % visible specific argument position
2819         \__stex_terms_custom_arg:wn
2820       } {
2821         % invisible
2822         \peek_charcode_remove:NTF * {
2823           % invisible specific argument position
2824           \__stex_terms_custom_arg_inv:wn
2825         } {
2826           % invisible next argument
2827           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2828         }
2829       }
2830     } {
2831       % next normal argument
2832       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2833     }
2834   }
2835 }
2836 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2837 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2838   \bool_set_true:N \l_tmpa_bool
2839   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2840 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2841 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2842   \str_set:Nx \l_tmpb_str {
2843     \str_item:Nn \l_tmpa_str { #1 }
2844   }
2845   \str_case:VnTF \l_tmpb_str {
2846     { X } {
2847       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2848     }
2849     { i } { \__stex_terms_custom_set_X:n { #1 } }
2850     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2851     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2852     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2853   }{}{
2854     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2855   }
2856
2857   \bool_if:nTF \l_tmpa_bool {
2858     \tl_put_right:Nx \l_tmpa_tl {
2859       \stex_annotate_invisible:n {
2860         \stex_term_arg:nn { \int_eval:n { #1 } }
2861         \exp_not:n { { #2 } }
2862       }
2863     }
2864   } {
2865     \tl_put_right:Nx \l_tmpa_tl {
2866       \stex_term_arg:nn { \int_eval:n { #1 } }
2867       \exp_not:n { { #2 } }
2868     }
2869   }
2870
2871   \__stex_terms_custom_loop:
2872 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2873 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2874   \str_set:Nx \l_tmpa_str {
2875     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2876     X
2877     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2878   }
2879 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2880 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2881   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2882   \__stex_terms_custom_loop:
2883 }

```

(End definition for __stex_terms_custom_component:.)

__stex_terms_custom_final:

```

2884 \cs_new_protected:Nn \__stex_terms_custom_final: {
2885   \int_compare:nNnTF \l_tmpb_int = 0 {
2886     \exp_args:Nnno \stex_term_oms:nnn
2887   }{
2888     \str_if_in:NnTF \l_tmpa_str {b} {
2889       \exp_args:Nnno \stex_term_ombind:nnn
2890     } {
2891       \exp_args:Nnno \stex_term_oma:nnn
2892     }
2893   }

```

```

2894 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2895 }

```

(End definition for `_stex_terms_custom_final:`)

`\symref`

`\symname`

```

2896 \NewDocumentCommand \symref { m m }{
2897   \let\compemph_uri_prev:\compemph@uri
2898   \let\compemph@uri\symrefemph@uri
2899   \STEXsymbol{#1}! [#2]
2900   \let\compemph@uri\compemph_uri_prev:
2901 }
2902
2903 \keys_define:nn { stex / symname } {
2904   post      .str_set_x:N    = \l_stex_symname_post_str
2905 }
2906
2907 \cs_new_protected:Nn \stex_symname_args:n {
2908   \str_clear:N \l_stex_symname_post_str
2909   \keys_set:nn { stex / symname } { #1 }
2910 }
2911
2912 \NewDocumentCommand \symname { O{} m }{
2913   \stex_symname_args:n { #1 }
2914   \stex_get_symbol:n { #2 }
2915   \str_set:Nx \l_tmpa_str {
2916     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2917   }
2918   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2919
2920   \let\compemph_uri_prev:\compemph@uri
2921   \let\compemph@uri\symrefemph@uri
2922   \exp_args:NNx \use:nn
2923   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2924     \l_tmpa_str \l_stex_symname_post_str
2925   ] }
2926   \let\compemph@uri\compemph_uri_prev:
2927 }

```

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

31.3 Notation Components

```

2928 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2929
2930 \str_new:N \l_stex_current_symbol_str
2931 \cs_new_protected:Nn \stex_highlight_term:nn {
2932   \exp_args:Nnx
2933   \use:nn {
2934     \str_set:Nx \l_stex_current_symbol_str { #1 }
2935     #2
2936   } {

```

```

2937 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2938 { \l_stex_current_symbol_str }
2939 }
2940 }
2941
2942 \cs_new_protected:Nn \stex_unhighlight_term:n {
2943 % \latexml_if:TF {
2944 % #1
2945 % } {
2946 % \rustex_if:TF {
2947 % #1
2948 % } {
2949 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2950 % }
2951 % }
2952 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2953 \cs_new_protected:Npn \comp #1 {
2954 \str_if_empty:NF \l_stex_current_symbol_str {
2955 \rustex_if:TF {
2956 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
2957 }{
2958 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2959 }
2960 }
2961 }
2962
2963 \cs_new_protected:Npn \compemph@uri #1 #2 {
2964 \compemph{ #1 }
2965 }
2966
2967
2968 \cs_new_protected:Npn \compemph #1 {
2969 #1
2970 }
2971
2972 \cs_new_protected:Npn \defemph@uri #1 #2 {
2973 \defemph{#1}
2974 }
2975
2976 \cs_new_protected:Npn \defemph #1 {
2977 \textbf{#1}
2978 }
2979
2980 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2981 \symrefemph{#1}
2982 }
2983
2984 \cs_new_protected:Npn \symrefemph #1 {
2985 \textbf{#1}
2986 }

```

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

`\ellipses`

```
2987 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2988 \bool_new:N \l_stex_inarray_bool
\parrayline 2989 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2990 \NewDocumentCommand \parray { m m } {
\parraycell 2991 \begin{group}
2992 \bool_set_true:N \l_stex_inarray_bool
2993 \begin{array}{#1}
2994 #2
2995 \end{array}
2996 \end{group}
2997 }
2998
2999 \NewDocumentCommand \prmatrix { m } {
3000 \begin{group}
3001 \bool_set_true:N \l_stex_inarray_bool
3002 \begin{matrix}
3003 #1
3004 \end{matrix}
3005 \end{group}
3006 }
3007
3008 \def \maybepline {
3009 \bool_if:NT \l_stex_inarray_bool {\hline}
3010 }
3011
3012 \def \parrayline #1 #2 {
3013 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3014 }
3015
3016 \def \pmrow #1 { \parrayline{}{ #1 } }
3017
3018 \def \parraylineh #1 #2 {
3019 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3020 }
3021
3022 \def \parraycell #1 {
3023 #1 \bool_if:NT \l_stex_inarray_bool {&}
3024 }

```

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

```
3025 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
3026 <*package>
3027
3028 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3029
3030 <@@=stex_features>
3031
3032 Warnings and error messages
3033 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3034   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3035 }
3036 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3037   Symbol~#1~not~assigned~in~interpretmodule~#2
3038 }
```

32.1 Imports with modification

```
3038 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3039   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3040     \__stex_features_get_symbol_from_cs:n { #1 }
3041   }{
3042     % argument is a string
3043     % is it a command name?
3044     \cs_if_exist:cTF { #1 }{
3045       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3046       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3047       \str_if_empty:NNTF \l_tmpa_str {
3048         \exp_args:Nx \cs_if_eq:NNTF {
3049           \tl_head:N \l_tmpa_tl
3050         } \stex_invoke_symbol:n {
3051           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3052         }{
3053           \__stex_features_get_symbol_from_string:n { #1 }
```

```

3054     }
3055   } {
3056     \__stex_features_get_symbol_from_string:n { #1 }
3057   }
3058   }{
3059     % argument is not a command name
3060     \__stex_features_get_symbol_from_string:n { #1 }
3061     % \l_stex_all_symbols_seq
3062   }
3063 }
3064 }
3065
3066 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3067   \str_set:Nn \l_tmpa_str { #1 }
3068   \bool_set_false:N \l_tmpa_bool
3069   \bool_if:NF \l_tmpa_bool {
3070     \tl_set:Nn \l_tmpa_tl {
3071       \msg_set:nnn{stex}{error/unknownsymbol}{
3072         No~symbol~#1~found!
3073       }
3074       \msg_error:nn{stex}{error/unknownsymbol}
3075     }
3076     \str_set:Nn \l_tmpa_str { #1 }
3077     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3078     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3079       \str_set:Nn \l_tmpb_str { ##1 }
3080       \str_if_eq:eeT { \l_tmpa_str } {
3081         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3082       } {
3083         \seq_map_break:n {
3084           \tl_set:Nn \l_tmpa_tl {
3085             \str_set:Nn \l_stex_get_symbol_uri_str {
3086               ##1
3087             }
3088             \__stex_features_get_symbol_check:
3089           }
3090         }
3091       }
3092     }
3093     \l_tmpa_tl
3094   }
3095 }
3096
3097 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3098   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3099   { \tl_tail:N \l_tmpa_tl }
3100   \tl_if_single:NTF \l_tmpa_tl {
3101     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3102       \exp_after:wN \str_set:Nn \exp_after:wN
3103       \l_stex_get_symbol_uri_str \l_tmpa_tl
3104       \__stex_features_get_symbol_check:
3105     }{
3106       % TODO
3107       % tail is not a single group

```

```

3108     }
3109 }{
3110     % TODO
3111     % tail is not a single group
3112 }
3113 }
3114
3115 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3116     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3117     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3118         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3119         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3120         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3121             \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3122                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3123             }
3124         }
3125     }{
3126         \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3127             \l_stex_current_copymodule_name_str~(inexplicably)
3128         }
3129     }
3130 }
3131
3132 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3133     \stex_import_module_uri:nn { #1 } { #2 }
3134     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3135     \stex_import_require_module:nnnn
3136     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3137     { \l_stex_import_path_str } { \l_stex_import_name_str }
3138     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3139     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3140     \seq_clear:N \l__stex_features_copymodule_fields_seq
3141     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3142         \seq_map_inline:cn {c_stex_module_###1_constants}{
3143             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3144                 ###1 ? ####1
3145             }
3146         }
3147     }
3148     \seq_clear:N \l_tmpa_seq
3149     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3150         name      = \l_stex_current_copymodule_name_str ,
3151         module     = \l_stex_current_module_str ,
3152         from       = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3153         includes   = \l_tmpa_seq ,
3154         fields     = \l_tmpa_seq
3155     }
3156     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3157         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3158     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3159     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3160     \stex_if_smsmode:TF {
3161         \stex_smsmode_set_codes:

```



```

3162 } {
3163   \begin{stex_annotate_env} {#4} {
3164     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3165   }
3166   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3167 }
3168 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3169 \bool_set_false:N \l_stex_html_do_output_bool
3170 }
3171 \cs_new_protected:Nn \stex_copymodule_end:n {
3172   \def \l_tmpa_cs ##1 ##2 {#1}
3173   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3174   \tl_clear:N \l_tmpa_tl
3175   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3176   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3177     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3178       \l_tmpa_cs{##1}{####1}
3179       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3180         \tl_put_right:Nx \l_tmpa_tl {
3181           \prop_set_from_keyval:cn {
3182             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3183           }{
3184             \exp_after:wN \prop_to_keyval:N \csname
3185               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3186             \endcsname
3187           }
3188           \seq_clear:c {
3189             l_stex_symdecl_
3190             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3191             _notations
3192           }
3193         }
3194         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3195         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3196         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3197           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3198           \tl_put_right:Nx \l_tmpa_tl {
3199             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3200               \stex_invoke_symbol:n {
3201                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3202               }
3203             }
3204           }
3205         }
3206       }{
3207         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3208         \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3209         \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3210         \tl_put_right:Nx \l_tmpa_tl {
3211           \prop_set_from_keyval:cn {
3212             l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3213           }{
3214             \prop_to_keyval:N \l_tmpa_prop
3215           }

```

```

3216         \seq_clear:c {
3217             l_stex_symdecl_
3218             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3219             _notations
3220         }
3221     }
3222     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3223     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3224         \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3225         \tl_put_right:Nx \l_tmpa_tl {
3226             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3227                 \stex_invoke_symbol:n {
3228                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3229                 }
3230             }
3231         }
3232     }
3233 }
3234 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3235     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3236 }
3237 % todo notations
3238 }}
3239 }
3240 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3241 \tl_put_left:Nx \l_tmpa_tl {
3242     \prop_set_from_keyval:cn {
3243         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3244     }{
3245         \prop_to_keyval:N \l_stex_current_copymodule_prop
3246     }
3247 }
3248 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3249 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3250 \exp_args:Nx \stex_do_aftergroup:n {
3251     \exp_args:No \exp_not:n \l_tmpa_tl
3252 }
3253 \stex_if_smsmode:F {
3254     \end{stex_annotate_env}
3255 }
3256 }
3257
3258 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3259     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3260     \stex_deactivate_macro:Nn \symdecl {module~environments}
3261     \stex_deactivate_macro:Nn \symdef {module~environments}
3262     \stex_deactivate_macro:Nn \notation {module~environments}
3263     \stex_reactivate_macro:N \assign
3264     \stex_reactivate_macro:N \renamedec1
3265     \stex_reactivate_macro:N \donotcopy
3266 }{
3267     \stex_copymodule_end:n {}
3268 }
3269

```

```

3270 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3271   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3272   \stex_deactivate_macro:Nn \symdecl {module~environments}
3273   \stex_deactivate_macro:Nn \symdef {module~environments}
3274   \stex_deactivate_macro:Nn \notation {module~environments}
3275   \stex_reactivate_macro:N \assign
3276   \stex_reactivate_macro:N \renamedekl
3277   \stex_reactivate_macro:N \donotcopy
3278 }{
3279   \stex_copymodule_end:n {
3280     \tl_if_exist:cF {
3281       l__stex_features_copymodule_##1?##2_def_tl
3282     }{
3283       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3284         ##1?##2
3285       }{\l_stex_current_copymodule_name_str}
3286     }
3287   }
3288 }
3289
3290 \NewDocumentCommand \donotcopy { 0{} m}{
3291   \stex_import_module_uri:nn { #1 } { #2 }
3292   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3293   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3294     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3295     \seq_map_inline:cn {c_stex_module_##1_constants}{
3296       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? #####1 }
3297       \bool_lazy_any_p:nT {
3298         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3299         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3300         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3301       }{
3302         % TODO throw error
3303       }
3304     }
3305   }
3306
3307   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3308   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3309   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3310 }
3311
3312 \NewDocumentCommand \assign { m m }{
3313   \stex_get_symbol_in_copymodule:n {#1}
3314   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3315   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3316 }
3317
3318 \keys_define:nn { stex / renamedekl } {
3319   name .str_set_x:N = \l_stex_renamedekl_name_str
3320 }
3321 \cs_new_protected:Nn \__stex_features_renamedekl_args:n {
3322   \str_clear:N \l_stex_renamedekl_name_str
3323

```

```

3324 \keys_set:nn { stex / renamedec1 } { #1 }
3325 }
3326
3327 \NewDocumentCommand \renamedec1 { 0{} m m }{
3328   \__stex_features_renamedec1_args:n { #1 }
3329   \stex_get_symbol_in_copymodule:n {#2}
3330   \stex_debug:nn{renamedec1}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3331   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3332   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3333     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3334       \l_stex_get_symbol_uri_str
3335     } }
3336   } {
3337     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3338     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3339     \prop_set_eq:cc {l_stex_symdecl_
3340       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3341     _prop
3342     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3343     \seq_set_eq:cc {l_stex_symdecl_
3344       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3345     _notations
3346     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3347     \prop_put:cnx {l_stex_symdecl_
3348       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3349     _prop
3350     }{ name }{ \l_stex_renamedec1_name_str }
3351     \prop_put:cnx {l_stex_symdecl_
3352       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3353     _prop
3354     }{ module }{ \l_stex_current_module_str }
3355     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3356       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3357     }
3358     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3359       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3360     } }
3361   }
3362 }
3363 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3364 % \_stex_notation_args:n { #1 }
3365 % \tl_clear:N \l_stex_symdecl_definiens_tl
3366 % \stex_get_symbol_in_copymodule:n { #2 }
3367 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3368 % % todo
3369 %}
3370 \stex_deactivate_macro:Nn \assign {copymodules}
3371 \stex_deactivate_macro:Nn \renamedec1 {copymodules}
3372 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3373
3374
3375 \seq_new:N \l_stex_implicit_morphisms_seq
3376 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3377   \stex_import_module_uri:nn { #1 } { #2 }

```

```

3378 \stex_debug:nn{implicits}{
3379   Implicit~morphism:~
3380   \l_stex_module_ns_str ? \l__stex_features_name_str
3381 }
3382 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3383   \l_stex_module_ns_str ? \l__stex_features_name_str
3384 }{
3385   \msg_error:nnn{stex}{error/conflictingmodules}{
3386     \l_stex_module_ns_str ? \l__stex_features_name_str
3387   }
3388 }
3389
3390 % TODO
3391
3392
3393
3394 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3395   \l_stex_module_ns_str ? \l__stex_features_name_str
3396 }
3397 }
3398

```

32.2 The feature environment

structural@feature

```

3399
3400 \NewDocumentEnvironment{structural@feature}{ m m m }{
3401   \stex_if_in_module:F {
3402     \msg_set:nnn{stex}{error/nomodule}{
3403       Structural~Feature~has~to~occur~in~a~module:\\
3404       Feature~#2~of~type~#1\\
3405       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3406     }
3407     \msg_error:nn{stex}{error/nomodule}
3408   }
3409
3410   \str_set:Nx \l_stex_module_name_str {
3411     \prop_item:Nn \l_stex_current_module_prop
3412     { name } / #2 - feature
3413   }
3414
3415   \str_set:Nx \l_stex_module_ns_str {
3416     \prop_item:Nn \l_stex_current_module_prop
3417     { ns }
3418   }
3419
3420
3421   \str_clear:N \l_tmpa_str
3422   \seq_clear:N \l_tmpa_seq
3423   \tl_clear:N \l_tmpa_tl
3424   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3425     origname = #2,
3426     name      = \l_stex_module_name_str ,
3427     ns        = \l_stex_module_ns_str ,

```

```

3428 imports = \exp_not:o { \l_tmpa_seq } ,
3429 constants = \exp_not:o { \l_tmpa_seq } ,
3430 content = \exp_not:o { \l_tmpa_tl } ,
3431 file = \exp_not:o { \g_stex_currentfile_seq } ,
3432 lang = \l_stex_module_lang_str ,
3433 sig = \l_tmpa_str ,
3434 meta = \l_tmpa_str ,
3435 feature = #1 ,
3436 }
3437
3438 \stex_if_smsmode:TF {
3439   \stex_smsmode_set_codes:
3440 } {
3441   \begin{stex_annotate_env}{ feature:#1 }{}
3442   \stex_annotate_invisible:nnn{header}{}{ #3 }
3443 }
3444 }{
3445   \str_set:Nx \l_tmpa_str {
3446     c_stex_feature_
3447     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3448     \prop_item:Nn \l_stex_current_module_prop { name }
3449     _prop
3450   }
3451   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3452   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3453   \stex_if_smsmode:TF {
3454     \exp_args:Nx \stex_add_to_sms:n {
3455       \prop_gset_from_keyval:cn {
3456         c_stex_feature_
3457         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3458         \prop_item:Nn \l_stex_current_module_prop { name }
3459         _prop
3460       } {
3461         origname = #2,
3462         name = \prop_item:cn { \l_tmpa_str } { name } ,
3463         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3464         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3465         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3466         content = \prop_item:cn { \l_tmpa_str } { content } ,
3467         file = \prop_item:cn { \l_tmpa_str } { file } ,
3468         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3469         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3470         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3471         feature = \prop_item:cn { \l_tmpa_str } { feature }
3472       }
3473     }
3474   } {
3475     \end{stex_annotate_env}
3476   }
3477 }
3478

```

32.3 Features

structure

```

3479
3480 \prop_new:N \l_stex_all_structures_prop
3481
3482 \keys_define:nn { stex / features / structure } {
3483   name .str_set_x:N = \l__stex_features_structure_name_str ,
3484 }
3485
3486 \cs_new_protected:Nn \__stex_features_structure_args:n {
3487   \str_clear:N \l__stex_features_structure_name_str
3488   \keys_set:nn { stex / features / structure } { #1 }
3489 }
3490
3491 %\stex_new_feature:nnnn { structure } { 0{} m } {
3492   % \__stex_features_structure_args:n { ##1 }
3493   % \str_if_empty:NT \l__stex_features_structure_name_str {
3494     % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3495   % }
3496 %} {
3497 %
3498 %}
3499
3500 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
3501   \__stex_features_structure_args:n { #1 }
3502   \str_if_empty:NT \l__stex_features_structure_name_str {
3503     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3504   }
3505   \exp_args:Nnnx
3506   \begin{structural@feature}{ structure }
3507     { \l__stex_features_structure_name_str }{}
3508     \seq_clear:N \l_tmpa_seq
3509     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3510
3511   }{
3512     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3513     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3514     \str_set:Nx \l_tmpa_str {
3515       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3516       \prop_item:Nn \l_stex_current_module_prop { name }
3517     }
3518     \seq_map_inline:Nn \l_tmpa_seq {
3519       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3520     }
3521     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3522     \exp_args:Nnx
3523     \AddToHookNext { env / mathstructure / after }{
3524       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3525         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3526       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3527     \STEXexport {
3528       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3529       { \prop_item:Nn \l_stex_current_module_prop { origname } }

```

```

3530         {\l_tmpa_str}
3531         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3532         {#2}{\l_tmpa_str}
3533     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3534     % \prop_item:Nn \l_stex_current_module_prop { origname },
3535     % \l_tmpa_str
3536     % }
3537     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3538     % #2,\l_tmpa_str
3539     % }
3540     % \tl_set:cx { #2 } {
3541     % \stex_invoke_structure:n { \l_tmpa_str }
3542     % }
3543 }
3544
3545 \end{structural@feature}
3546 % \g_stex_last_feature_prop
3547 }

```

\instantiate

```

3548 \seq_new:N \l__stex_features_structure_field_seq
3549 \str_new:N \l__stex_features_structure_field_str
3550 \str_new:N \l__stex_features_structure_def_tl
3551 \prop_new:N \l__stex_features_structure_prop
3552 \NewDocumentCommand \instantiate { m O{} m }{
3553     \stex_smsmode_set_codes:
3554     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3555     \prop_set_eq:Nc \l__stex_features_structure_prop {
3556         c_stex_feature_\l_tmpa_str _prop
3557     }
3558     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3559     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3560         \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3561         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3562             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3563             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3564             {!} \l_tmpa_tl
3565             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3566                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3567                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3568                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3569             }{
3570                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3571                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3572                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3573                 \l_tmpa_tl
3574                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3575                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3576                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3577                 }{
3578                     \tl_clear:N \l_tmpb_tl
3579                 }
3580             }
3581         }{

```



```

3582 \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3583 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3584   \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3585   \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3586   \tl_clear:N \l_tmpa_tl
3587 }{
3588   % TODO throw error
3589 }
3590 }
3591 % \l_tmpa_str: name
3592 % \l_tmpa_tl: definiens
3593 % \l_tmpb_tl: notation
3594 \tl_if_empty:NT \l__stex_features_structure_field_str {
3595   % TODO throw error
3596 }
3597 \str_clear:N \l_tmpb_str
3598
3599 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3600 \seq_map_inline:Nn \l_tmpa_seq {
3601   \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3602   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3603   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3604     \seq_map_break:n {
3605       \str_set:Nn \l_tmpb_str { ###1 }
3606     }
3607   }
3608 }
3609 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3610 \l_tmpb_str
3611
3612 \tl_if_empty:NTF \l_tmpb_tl {
3613   \tl_if_empty:NF \l_tmpa_tl {
3614     \exp_args:Nx \use:n {
3615       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3616     }
3617   }
3618 }{
3619   \tl_if_empty:NTF \l_tmpa_tl {
3620     \exp_args:Nx \use:n {
3621       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3622     }
3623   }
3624 }{
3625   \exp_args:Nx \use:n {
3626     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3627     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3628   }
3629 }
3630 }
3631 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3632 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3633 % #3/\l__stex_features_structure_field_str
3634 % \par
3635 % \expandafter\present\csname

```

```

3636 %      l_stex_symdecl_
3637 %      \prop_item:Nn \l_stex_current_module_prop {ns} ?
3638 %      \prop_item:Nn \l_stex_current_module_prop {name} ?
3639 %      #3/\l__stex_features_structure_field_str
3640 %      _prop
3641 %      \endcsname
3642 }
3643
3644 \tl_clear:N \l__stex_features_structure_def_tl
3645
3646 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3647 \seq_map_inline:Nn \l_tmpa_seq {
3648   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3649   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3650   \exp_args:Nx \use:n {
3651     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3652
3653     }
3654   }
3655
3656   \prop_if_exist:cF {
3657     l_stex_symdecl_
3658     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3659     \prop_item:Nn \l_stex_current_module_prop {name} ?
3660     #3/\l_tmpa_str
3661     _prop
3662   }{
3663     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3664     \l_tmpb_str
3665     \exp_args:Nx \use:n {
3666       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3667     }
3668   }
3669 }
3670
3671 \symdecl*[type={\STEXsymbol{module-type}}{
3672   \_stex_term_math_oms:nnnn {
3673     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3674     \prop_item:Nn \l__stex_features_structure_prop {name}
3675     }{}{0}{}
3676   }{}{#3}
3677
3678 % TODO: -> sms file
3679
3680 \tl_set:cx{ #3 }{
3681   \stex_invoke_structure:nnn {
3682     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3683     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3684   } {
3685     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3686     \prop_item:Nn \l__stex_features_structure_prop {name}
3687   }
3688 }
3689

```

```
3690 }
```

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

```
\stex_invoke_structure:nnn
```

```
3691 % #1: URI of the instance
3692 % #2: URI of the instantiated module
3693 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3694   \tl_if_empty:nTF{ #3 }{
3695     \prop_set_eq:Nc \l__stex_features_structure_prop {
3696       c_stex_feature_ #2 _prop
3697     }
3698     \tl_clear:N \l_tmpa_tl
3699     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3700     \seq_map_inline:Nn \l_tmpa_seq {
3701       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3702       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3703       \cs_if_exist:cT {
3704         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3705       }{
3706         \tl_if_empty:NF \l_tmpa_tl {
3707           \tl_put_right:Nn \l_tmpa_tl {,}
3708         }
3709         \tl_put_right:Nx \l_tmpa_tl {
3710           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3711         }
3712       }
3713     }
3714     \exp_args:No \mathstrut \l_tmpa_tl
3715   }{
3716     \stex_invoke_symbol:n{#1/#3}
3717   }
3718 }
```

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

```
3719 \endpackage
```

Chapter 33

STEX -Statements Implementation

```
3720 <*package>
3721
3722 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3723
3724 \protected\def\ignorespacesandpars{
3725   \begingroup\catcode13=10\relax
3726   \@ifnextchar\par{
3727     \endgroup\expandafter\ignorespacesandpars\@gobble
3728   }{
3729     \endgroup
3730   }
3731 }
3732
3733 <@@=stex_statements>
3734
3735 Warnings and error messages
```

\titleemph

```
3735 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3736 \keys_define:nn {stex / definiendum }{
3737   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3738   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3739   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3740 }
3741 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3742   \str_clear:N \l__stex_statements_definiendum_root_str
3743   \tl_clear:N \l__stex_statements_definiendum_post_tl
3744   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3745 \keys_set:nn { stex / definiendum }{ #1 }
3746 }
3747 \NewDocumentCommand \definiendum { 0{ } m m } {
3748   \__stex_statements_definiendum_args:n { #1 }
3749   \stex_get_symbol:n { #2 }
3750   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3751   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3752     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3753       \tl_set:Nn \l_tmpa_tl { #3 }
3754     } {
3755       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3756       \tl_set:Nn \l_tmpa_tl {
3757         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3758       }
3759     }
3760   } {
3761     \tl_set:Nn \l_tmpa_tl { #3 }
3762   }
3763
3764   % TODO root
3765   \rustex_if:TF {
3766     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3767   } {
3768     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3769   }
3770 }
3771 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3772 \NewDocumentCommand \definame { 0{ } m } {
3773   \__stex_statements_definiendum_args:n { #1 }
3774   % TODO: root
3775   \stex_get_symbol:n { #2 }
3776   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3777   \str_set:Nx \l_tmpa_str {
3778     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3779   }
3780   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3781   \rustex_if:TF {
3782     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3783       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3784     }
3785   } {
3786     \defemph@uri {
3787       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3788     } { \l_stex_get_symbol_uri_str }
3789   }
3790 }
3791 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3792
3793 \keys_define:nn {stex / sdefinition }{
3794   type      .str_set_x:N = \sdefinitiontype,
3795   id        .str_set_x:N = \sdefinitionid,
3796   title     .tl_set:N    = \sdefinitiontitle
3797 }
3798 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3799   \str_clear:N \sdefinitiontype
3800   \str_clear:N \sdefinitionid
3801   \tl_clear:N \sdefinitiontitle
3802   \keys_set:nn { stex / sdefinition }{ #1 }
3803 }
3804
3805 \NewDocumentEnvironment{sdefinition}{0{}}{
3806   \__stex_statements_sdefinition_args:n{ #1 }
3807   \stex_reactivate_macro:N \definiendum
3808   \stex_reactivate_macro:N \definame
3809   \stex_smsmode_set_codes:
3810   \stex_if_smsmode:F {
3811     \exp_args:Nnnx
3812     \begin{stex_annotate_env}{definition}{}
3813     \str_if_empty:NF \sdefinitiontype {
3814       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3815     }
3816   }
3817   \clist_set:No \l_tmpa_clist \sdefinitiontype
3818   \tl_clear:N \l_tmpa_tl
3819   \clist_map_inline:Nn \l_tmpa_clist {
3820     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3821       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3822     }
3823   }
3824   \tl_if_empty:NTF \l_tmpa_tl {
3825     \__stex_statements_sdefinition_start:
3826   }{
3827     \l_tmpa_tl
3828   }
3829   \stex_ref_new_doc_target:n \sdefinitionid
3830 }{
3831   \clist_set:No \l_tmpa_clist \sdefinitiontype
3832   \tl_clear:N \l_tmpa_tl
3833   \clist_map_inline:Nn \l_tmpa_clist {
3834     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3835       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3836     }
3837   }
3838   \tl_if_empty:NTF \l_tmpa_tl {
3839     \__stex_statements_sdefinition_end:
3840   }{
3841     \l_tmpa_tl
3842   }
3843   \stex_if_smsmode:F {
3844     \end{stex_annotate_env}

```

```

3845 }
3846 }

```

`\stexpatchdefinition`

```

3847 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3848   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3849     ~(\sdefinitiontitle)
3850   }~}
3851 }
3852 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3853
3854 \newcommand\stexpatchdefinition[3] [] {
3855   \str_set:Nx \l_tmpa_str{ #1 }
3856   \str_if_empty:NTF \l_tmpa_str {
3857     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3858     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3859   }{
3860     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3861     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3862   }
3863 }

```

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

`\inlinedef inline:`

```

3864 \NewDocumentCommand \inlinedef { m } {
3865   \begingroup
3866   \stex_reactivate_macro:N \definiendum
3867   \stex_reactivate_macro:N \definame
3868   \stex_ref_new_doc_target:n{
3869     #1
3870   }
3871 }

```

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

33.2 Assertions

`sassertion`

```

3872
3873 \keys_define:nn {stex / sassertion }{
3874   type      .str_set_x:N = \sassertiontype,
3875   id        .str_set_x:N = \sassertionid,
3876   title     .tl_set:N     = \sassertiontitle ,
3877   name      .str_set_x:N = \sassertionname
3878 }
3879 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3880   \str_clear:N \sassertiontype
3881   \str_clear:N \sassertionid
3882   \str_clear:N \sassertionname
3883   \tl_clear:N \sassertiontitle
3884   \keys_set:nn { stex / sassertion }{ #1 }
3885 }

```

```

3886
3887 %\tl_new:N \g__stex_statements_aftergroup_tl
3888
3889 \NewDocumentEnvironment{sassertion}{0{}}{
3890   \__stex_statements_sassertion_args:n{ #1 }
3891   \stex_smsmode_set_codes:
3892   \stex_if_smsmode:F {
3893     \exp_args:Nnnx
3894     \begin{stex_annotate_env}{assertion}{}
3895     \str_if_empty:NF \sassertiontype {
3896       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3897     }
3898   }
3899   \clist_set:N \l_tmpa_clist \sassertiontype
3900   \tl_clear:N \l_tmpa_tl
3901   \clist_map_inline:Nn \l_tmpa_clist {
3902     \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
3903       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
3904     }
3905   }
3906   \tl_if_empty:NTF \l_tmpa_tl {
3907     \__stex_statements_sassertion_start:
3908   }{
3909     \l_tmpa_tl
3910   }
3911   \stex_ref_new_doc_target:n \sassertionid
3912 }{
3913   \clist_set:N \l_tmpa_clist \sassertiontype
3914   \tl_clear:N \l_tmpa_tl
3915   \clist_map_inline:Nn \l_tmpa_clist {
3916     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
3917       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
3918     }
3919   }
3920   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
3921   \tl_if_empty:NTF \l_tmpa_tl {
3922     \__stex_statements_sassertion_end:
3923   }{
3924     \l_tmpa_tl
3925   }
3926   \stex_if_smsmode:F {
3927     \end{stex_annotate_env}
3928   }
3929 }

```

\stexpatchassertion

```

3930
3931 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3932   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
3933     (\sassertiontitle)
3934   }~}
3935 }
3936 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3937

```



```

3938 \newcommand\stexpatchassertion[3] [] {
3939   \str_set:Nx \l_tmpa_str{ #1 }
3940   \str_if_empty:NTF \l_tmpa_str {
3941     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3942     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3943   }{
3944     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3945     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3946   }
3947 }

```

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

\inlineass inline:

```

3948 \NewDocumentCommand \inlineass { m } {
3949   \begingroup
3950   \stex_ref_new_doc_target:n{
3951     #1
3952   \endgroup
3953 }

```

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

33.3 Examples

sexample

```

3954
3955 \keys_define:nn {stex / sexample }{
3956   type      .str_set_x:N = \exampletype,
3957   id        .str_set_x:N = \sexampleid,
3958   title     .tl_set:N     = \sexampletitle,
3959   for       .clist_set:N  = \sexamplefor,
3960 }
3961 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3962   \str_clear:N \sexampletype
3963   \str_clear:N \sexampleid
3964   \tl_clear:N \sexampletitle
3965   \clist_clear:N \sexamplefor
3966   \keys_set:nn { stex / sexample }{ #1 }
3967 }
3968
3969 \NewDocumentEnvironment{sexample}{0{}}{
3970   \__stex_statements_sexample_args:n{ #1 }
3971   \stex_smsmode_set_codes:
3972   \stex_if_smsmode:F {
3973     \seq_clear:N \l_tmpa_seq
3974     \clist_map_inline:Nn \sexamplefor {
3975       \str_if_eq:nnF{ ##1 }{}{
3976         \stex_get_symbol:n { ##1 }
3977         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3978           \l_stex_get_symbol_uri_str
3979         }
3980       }

```

```

3981   }
3982   \exp_args:Nnnx
3983   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3984   \str_if_empty:NF \sexamplotype {
3985     \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
3986   }
3987 }
3988 \stex_ref_new_doc_target:n \sexampleid
3989 \clist_set:No \l_tmpa_clist \sexamplotype
3990 \tl_clear:N \l_tmpa_tl
3991 \clist_map_inline:Nn \l_tmpa_clist {
3992   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
3993     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3994   }
3995 }
3996 \tl_if_empty:NTF \l_tmpa_tl {
3997   \__stex_statements_sexample_start:
3998 }{
3999   \l_tmpa_tl
4000 }
4001 }{
4002   \clist_set:No \l_tmpa_clist \sexamplotype
4003   \tl_clear:N \l_tmpa_tl
4004   \clist_map_inline:Nn \l_tmpa_clist {
4005     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4006       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4007     }
4008   }
4009   \tl_if_empty:NTF \l_tmpa_tl {
4010     \__stex_statements_sexample_end:
4011   }{
4012     \l_tmpa_tl
4013   }
4014   \stex_if_smsmode:F {
4015     \end{stex_annotate_env}
4016   }
4017 }

```

\stexpatchexample

```

4018
4019 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4020   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
4021     (\sexamplotype)
4022   }~}
4023 }
4024 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4025
4026 \newcommand\stexpatchexample[3]{} {
4027   \str_set:Nx \l_tmpa_str{ #1 }
4028   \str_if_empty:NTF \l_tmpa_str {
4029     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4030     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4031   }{
4032     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }

```

```

4033     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4034   }
4035 }

```

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

`\inlineex` inline:

```

4036 \NewDocumentCommand \inlineex { m } {
4037   \begingroup
4038   \stex_ref_new_doc_target:n{
4039     #1
4040   \endgroup
4041 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4042 \keys_define:nn { stex / sparagraph } {
4043   id      .str_set_x:N = \sparagraphid ,
4044   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4045   type    .str_set_x:N = \sparagraphtype ,
4046   for     .str_set_x:N = \sparagraphfor ,
4047   from    .tl_set_x:N  = \sparagraphfrom ,
4048   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4049   name    .str_set:N   = \sparagraphname
4050 }
4051
4052 \cs_new_protected:Nn \stex_sparagraph_args:n {
4053   \tl_clear:N \l_stex_sparagraph_title_tl
4054   \tl_clear:N \sparagraphfrom
4055   \tl_clear:N \l_stex_sparagraph_start_tl
4056   \str_clear:N \sparagraphid
4057   \str_clear:N \sparagraphtype
4058   \str_clear:N \sparagraphfor
4059   \str_clear:N \sparagraphname
4060   \keys_set:nn { stex / sparagraph }{ #1 }
4061 }
4062 \newif\if@in@omtext\@in@omtextfalse
4063
4064 \NewDocumentEnvironment {sparagraph} { 0{} } {
4065   \stex_sparagraph_args:n { #1 }
4066   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4067     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4068   }{
4069     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4070   }
4071   \@in@omtexttrue
4072   \stex_smsmode_set_codes:
4073   \stex_if_smsmode:F {
4074     \exp_args:Nnnx
4075     \begin{stex_annotate_env}{paragraph}{}

```

```

4076 \str_if_empty:NF \sparagraphtype {
4077 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4078 }
4079 }
4080 \clist_set:Nn \l_tmpa_clist \sparagraphtype
4081 \tl_clear:N \l_tmpa_tl
4082 \clist_map_inline:Nn \l_tmpa_clist {
4083 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4084 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4085 }
4086 }
4087 \tl_if_empty:NTF \l_tmpa_tl {
4088 \__stex_statements_sparagraph_start:
4089 }{
4090 \l_tmpa_tl
4091 }
4092 \stex_ref_new_doc_target:n \sparagraphid
4093 \ignorespacesandpars
4094 }{
4095 \clist_set:Nn \l_tmpa_clist \sparagraphtype
4096 \tl_clear:N \l_tmpa_tl
4097 \clist_map_inline:Nn \l_tmpa_clist {
4098 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4099 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4100 }
4101 }
4102 \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4103 \tl_if_empty:NTF \l_tmpa_tl {
4104 \__stex_statements_sparagraph_end:
4105 }{
4106 \l_tmpa_tl
4107 }
4108 \stex_if_smsmode:F {
4109 \end{stex_annotate_env}
4110 }
4111 }

```

\stexpatchparagraph

```

4112
4113 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4114 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4115 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4116 \titleemph{\l_stex_sparagraph_title_tl}:~
4117 }
4118 }{
4119 \titleemph{\l_stex_sparagraph_start_tl}~
4120 }
4121 }
4122 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4123
4124 \newcommand\stexpatchparagraph[3] [] {
4125 \str_set:Nx \l_tmpa_str{ #1 }
4126 \str_if_empty:NTF \l_tmpa_str {
4127 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }

```

```

4128     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4129   }{
4130     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4131     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4132   }
4133 }

```

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

symboldoc

```

4134 \NewDocumentEnvironment{symboldoc}{ m }{
4135   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4136   \seq_clear:N \l_tmpb_seq
4137   \seq_map_inline:Nn \l_tmpa_seq {
4138     \str_if_eq:nnF{ ##1 }{}{
4139       \stex_get_symbol:n { ##1 }
4140       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4141         \l_stex_get_symbol_uri_str
4142       }
4143     }
4144   }
4145   \par
4146   \exp_args:Nnnx
4147   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4148 }{
4149   \end{stex_annotate_env}
4150 }
4151 \</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

```
4152 <*package>
4153 <@@=stex_sproof>
4154
4155 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4156
```

34.2 Proofs

We first define some keys for the proof environment.

```
4157 \keys_define:nn { stex / spf } {
4158   id          .str_set:N = \l__stex_sproof_spf_id_str,
4159   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4160   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4161   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4162   proofend   .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4163   type       .tl_set:N  = \l__stex_sproof_spf_type_tl,
4164   title      .tl_set:N  = \l__stex_sproof_spf_title_tl,
4165   continues  .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4166   functions  .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4167   method     .tl_set:N  = \l__stex_sproof_spf_method_tl
4168 }
4169 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4170   \str_clear:N \l__stex_sproof_spf_id_str
4171   \tl_clear:N \l__stex_sproof_spf_display_tl
4172   \tl_clear:N \l__stex_sproof_spf_for_tl
4173   \tl_clear:N \l__stex_sproof_spf_from_tl
4174   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4175   \tl_clear:N \l__stex_sproof_spf_type_tl
4176   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4177 \tl_clear:N \l__stex_sproof_spf_continues_tl
4178 \tl_clear:N \l__stex_sproof_spf_functions_tl
4179 \tl_clear:N \l__stex_sproof_spf_method_tl
4180 \keys_set:nn { stex / spf }{ #1 }
4181 }

```

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

```

4182 \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.

```

4183 \newcount\count_ten
4184 \newenvironment{pst@with@label}[1]{
4185   \edef\pst@label{#1}
4186   \advance\count_ten by 1\relax
4187   \count_ten=1
4188 }{
4189   \advance\count_ten by -1\relax
4190 }

```

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

```

4191 \def\the@pst@label{
4192   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4193 }

```

(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.

```

4194 \keys_define:nn { stex / pstlabel }{
4195   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4196   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4197   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4198 }
4199 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4200 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4201 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4202 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4203 }
4204 \__stex_sproof_pstlabel_args:n {}
4205 \newcommand\setpstlabelstyle[1]{
4206   \__stex_sproof_pstlabel_args:n {#1}
4207 }
4208 \newcommand\setpstlabelstyledefault{%
4209   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4210 }

```

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

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

```

4211 \ExplSyntaxOff
4212 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4213 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4214 \def\pst@make@label@short#1#2{#2}
4215 \def\pst@make@label@empty#1#2{}
4216 \ExplSyntaxOn
4217 \def\pstlabelstyle#1{%
4218   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4219 }%
4220 \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.

```

4221 \def\next@pst@label{%
4222   \global\advance\count\count10 by 1%
4223 }%

```

(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

```

4224 \def\sproof@box{
4225   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4226 }
4227 \def\spf@proofend{\sproof@box}
4228 \def\sproofend{
4229   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4230     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4231   }
4232 }
4233 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4234 \def\spf@proofsketch@kw{Proof Sketch}
4235 \def\spf@proof@kw{Proof}
4236 \def\spf@step@kw{Step}

```


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

For the other languages, we set up triggers

```

4237 \AddToHook{begindocument}{
4238   \ltx@ifpackageloaded{babel}{
4239     \makeatletter
4240     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4241     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4242       \input{sproof-ngerman.ldf}
4243     }
4244     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4245       \input{sproof-finnish.ldf}
4246     }
4247     \clist_if_in:NnT \l_tmpa_clist {french}{
4248       \input{sproof-french.ldf}
4249     }
4250     \clist_if_in:NnT \l_tmpa_clist {russian}{
4251       \input{sproof-russian.ldf}
4252     }
4253     \makeatother
4254   }
4255 }

```

`spfsketch`

```

4256 \newcommand\spfsketch[2][]{
4257   \__stex_sproof_spf_args:n{#1}
4258   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4259     \titleemph{
4260       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4261         \spf@proofsketch@kw
4262       }{
4263         \l__stex_sproof_spf_type_tl
4264       }
4265     }:
4266   }
4267   {~#2}
4268   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4269   \sproofend
4270 }

```

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

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

```

4271 \newenvironment{spfeq}[2][]{
4272   \__stex_sproof_spf_args:n{#1}
4273   %\sref@target
4274   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4275     \titleemph{
4276       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4277         \spf@proof@kw
4278       }{

```

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

¹⁵EDNOTE: document above

```

4279         \l__stex_sproof_spf_type_tl
4280     }
4281     }:
4282 }
4283 {-#2}
4284 \begin{displaymath}\begin{array}{rcll}
4285 }{
4286 \end{array}\end{displaymath}
4287 }

```

(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.

```

4288 \newenvironment{spf@proof}[2] []{
4289   \l__stex_sproof_spf_args:n{#1}
4290   %\sref@target
4291   \count_ten=10
4292   \par\noindent
4293   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4294     \titleemph{
4295       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4296         \spf@proof@kw
4297       }{
4298         \l__stex_sproof_spf_type_tl
4299       }
4300     }:
4301   }
4302   {-#2}
4303   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4304   \def\pst@label{}
4305   \newcount\pst@count% initialize the labeling mechanism
4306   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4307 }{
4308 \end{pst@with@label}\end{description}
4309 }
4310 \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4311 \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4312 \newcommand\spfidea[2] []{
4313   \l__stex_sproof_spf_args:n{#1}
4314   \titleemph{
4315     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4316       \l__stex_sproof_spf_type_tl
4317     }:
4318   }-#2
4319   \sproofend
4320 }

```

(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
4321 \newenvironment{spfstep}[1][]{
4322   \_stex_sproof_spf_args:n{#1}
4323   \@in@omtexttrue
4324   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4325     \item[\the@pst@label]
4326   }
4327   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4328     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4329   }
4330   %\sref@label{id{\pst@label}
4331   \ignorespacesandpars
4332 }{
4333   \next@pst@label\ignorespacesandpars
4334 }

```

sproofcomment

```

4335 \newenvironment{sproofcomment}[1][]{
4336   \_stex_sproof_spf_args:n{#1}
4337   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4338     \item[\the@pst@label]
4339   }
4340 }{
4341   \next@pst@label
4342 }

```

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.

```

4343 \newenvironment{subproof}[2][]{
4344   \_stex_sproof_spf_args:n{#1}
4345   \def\@test{#2}
4346   \ifx\@test\empty\else
4347     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4348       \item[\the@pst@label]
4349     }{#2}
4350   \fi
4351   \begin{pst@with@label}{\pst@label,\number\count_ten}
4352 }{
4353   \end{pst@with@label}\next@pst@label
4354 }

```

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

```

4355 \newenvironment{spfcases}[2][]{
4356   \def\@test{#1}
4357   \ifx\@test\empty
4358     \begin{subproof}[method=by-cases]{#2}

```

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

```

4359 \else
4360   \begin{subproof}[#1,method=by-cases]{#2}
4361 \fi
4362 }{
4363   \end{subproof}
4364 }

```

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

```

4365 \newenvironment{spfcase}[2] [] {
4366   \__stex_sproof_spf_args:n{#1}
4367   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4368     \item[\the@pst@label]
4369   }
4370   \def\@test{#2}
4371   \ifx\@test\@empty
4372   \else
4373     {\titleemph{#2}:~}
4374   \fi
4375   \begin{pst@with@label}{\pst@label,\number\count_ten}
4376 }{
4377   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4378     \sproofend
4379   }
4380   \end{pst@with@label}
4381   \next@pst@label
4382 }

```

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

```

4383 \newcommand\spfcasesketch[3] [] {
4384   \__stex_sproof_spf_args:n{#1}
4385   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4386     \item[\the@pst@label]
4387   }
4388   \def\@test{#2}
4389   \ifx\@test\@empty
4390   \else
4391     {\titleemph{#2}:~}
4392   \fi#3
4393   \next@pst@label
4394 }%

```

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.

```

4395 \keys_define:nn { stex / just }{
4396   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4397   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4398   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4399   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4400 }

```

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

`justification`

4401 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4402 `\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.

4403 `\newcommand\justarg[2] [] {#2}`

4404 `\</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

```
4405 <*package>
4406
4407 %%%%%%%%%% others.dtx %%%%%%%%%%
4408
4409 <@@=stex_others>
    Warnings and error messages
4410 % None

\MSC Math subject classifier

4411 \NewDocumentCommand \MSC {m} {
4412 % TODO
4413 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4414 \@ifpackageloaded{tikzinput}{
4415 \RequirePackage{stex-tikzinput}
4416 }{}
4417 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4418 \*package>
4419 \@@=stex_modules>
4420
4421 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4422
4423 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4424 \begingroup
4425 \stex_module_setup:nn{
4426   ns=\c_stex_metatheory_ns_str,
4427   meta=NONE
4428 }{Metatheory}
4429 \stex_reactivate_macro:N \symdecl
4430 \stex_reactivate_macro:N \notation
4431 \stex_reactivate_macro:N \symdef
4432 \ExplSyntaxOff
4433 \csname stex_suppress_html:n\endcsname{
4434   % is-a (a:A, a \in A, a is an A, etc.)
4435   \symdecl[args=ai]{isa}
4436   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4437   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4438   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4439
4440   % bind (\forall, \Pi, \lambda etc.)
4441   \symdecl[args=Bi]{bind}
4442   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4443   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4444   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4445
4446   % dummy variable
4447   \symdecl{dummyvar}
4448   \notation[underscore]{dummyvar}{\comp\_}
4449   \notation[dot]{dummyvar}{\comp\cdot}
4450   \notation[dash]{dummyvar}{\comp{\rm --}}
4451
4452   %fromto (function space, Hom-set, implication etc.)
```

```

4453 \symdecl[args=ai]{fromto}
4454 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4455 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4456
4457 % mapto (lambda etc.)
4458 %\symdecl[args=Bi]{mapto}
4459 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4460 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4461 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4462
4463 % function/operator application
4464 \symdecl[args=ia]{apply}
4465 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4466 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4467
4468 % ‘type’ of all collections (sets, classes, types, kinds)
4469 \symdecl{collection}
4470 \notation[U]{collection}{\comp{\mathcal{U}}{}}
4471 \notation[set]{collection}{\comp{\textsf{Set}}{}}
4472
4473 % sequences
4474 \symdecl[args=1]{seqtype}
4475 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4476
4477 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4478 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4479
4480 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4481 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4482 % ^ superceded by \aseqfromto and \livar/\uivar
4483
4484 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4485 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4486 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4487
4488 % letin (‘let’, local definitions, variable substitution)
4489 \symdecl[args=bii]{letin}
4490 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4491 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4492 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4493
4494 % structures
4495 \symdecl*[args=1]{module-type}
4496 \notation{module-type}{\mathtt{MOD} #1}
4497 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4498 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4499
4500 }
4501 \ExplSyntaxOn
4502 \stex_add_to_current_module:n{
4503   \let\nappa\apply
4504   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4505   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4506   \def\livar{\csname sequence-index\endcsname[li]}

```



```

4507 \def\uivar{\csname sequence-index\endcsname[ui]}
4508 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4509 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4510 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4511 }
4512 \__stex_modules_end_module:
4513 \endgroup
4514 \endpackage

```

Chapter 37

Tikzinput Implementation

```
4515 <*package>
4516
4517 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4518
4519 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4520 \RequirePackage{l3keys2e}
4521
4522 \keys_define:nn { tikzinput } {
4523   image .bool_set:N = \c_tikzinput_image_bool,
4524   image .default:n = false ,
4525   unknown .code:n = {}
4526 }
4527
4528 \ProcessKeysOptions { tikzinput }
4529
4530 \bool_if:NTF \c_tikzinput_image_bool {
4531   \RequirePackage{graphicx}
4532
4533   \providecommand\usetikzlibrary[]{}
4534   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4535 }{
4536   \RequirePackage{tikz}
4537   \RequirePackage{standalone}
4538
4539   \newcommand \tikzinput [2] [] {
4540     \setkeys{Gin}{#1}
4541     \ifx \Gin@ewidth \Gin@exclamation
4542       \ifx \Gin@eheight \Gin@exclamation
4543         \input { #2 }
4544       \else
4545         \resizebox{!}{ \Gin@eheight }{
4546           \input { #2 }
4547         }
4548       \fi
4549     \else
4550       \ifx \Gin@eheight \Gin@exclamation
4551         \resizebox{ \Gin@ewidth }{!}{
4552           \input { #2 }
```

```

4553     }
4554     \else
4555         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4556             \input { #2 }
4557         }
4558     \fi
4559 \fi
4560 }
4561 }
4562
4563 \newcommand \ctikzinput [2] [] {
4564     \begin{center}
4565         \tikzinput [1] {#2}
4566     \end{center}
4567 }
4568
4569 \@ifpackageloaded{stex}{
4570     \RequirePackage{stex-tikzinput}
4571 }{}
4572
4573 </package>
4574 <*stex>
4575 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4576 \RequirePackage{stex}
4577 \RequirePackage{tikzinput}
4578
4579 \newcommand\mhtikzinput [2] [] {%
4580     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4581     \stex_in_repository:nn\Gin@mhrepos{
4582         \tikzinput [1]{\mhpath{##1}{#2}}
4583     }
4584 }
4585 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4586 </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.

```
4587 \*cls)
4588 \@@=document_structure)
4589 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4590 \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`

```
4591 \keys_define:nn{ document-structure / pkg }{
4592   class      .str_set_x:N = \c_document_structure_class_str,
4593   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4594   report     .code:n      = {
4595     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4596     \str_set:Nn \c_document_structure_class_str {report}
4597   },
4598   book       .code:n      = {
4599     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4600     \str_set:Nn \c_document_structure_class_str {book}
4601   },
4602   bookpart   .code:n      = {
4603     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4604     \str_set:Nn \c_document_structure_class_str {book}
4605     \str_set:Nn \c_document_structure_topsect_str {chapter}
4606   },
```

```

4607 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4608 unknown     .code:n      = {
4609   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4610 }
4611 }
4612 \ProcessKeysOptions{ document-structure / pkg }
4613 \str_if_empty:NT \c_document_structure_class_str {
4614   \str_set:Nn \c_document_structure_class_str {article}
4615 }
4616 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4617   {\c_document_structure_class_str}
4618

```

38.3 Beefing up the document environment

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

```

4619 \RequirePackage{omdoc}
4620 \bool_if:NF \c_document_structure_minimal_bool {
4621   \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

```

4622 \keys_define:nn { document-structure / document }{
4623   id .str_set_x:N = \c_document_structure_document_id_str
4624 }
4625 \let\__document_structure_orig_document=\document
4626 \renewcommand{\document}[1][]{
4627   \keys_set:nn{ document-structure / document }{ #1 }
4628   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4629   \__document_structure_orig_document
4630 }

```

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

```

4631 }
4632 \</cls>

```

38.4 Implementation: OMDoc Package

```

4633 \<*package>
4634 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4635 \RequirePackage{expl-keys-tr-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

```

4636
4637 \keys_define:nn{ document-structure / pkg }{
4638   class      .str_set_x:N = \c_document_structure_class_str,
4639   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4640   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4641 }
4642 \ProcessKeysOptions{ document-structure / pkg }
4643 \str_if_empty:NT \c_document_structure_class_str {
4644   \str_set:Nn \c_document_structure_class_str {article}
4645 }
4646 \str_if_empty:NT \c_document_structure_topsect_str {
4647   \str_set:Nn \c_document_structure_topsect_str {section}
4648 }

```

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

```

4649 \RequirePackage{xspace}
4650 \RequirePackage{comment}
4651 \AddToHook{begindocument}{
4652   \ltx@ifpackageloaded{babel}{
4653     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4654     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4655       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4656     }
4657   }{}
4658 }

```

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

```

4659 %\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.

```

4660 \int_new:N \l_document_structure_section_level_int
4661 \str_case:VnF \c_document_structure_topsect_str {
4662   {part}{
4663     \int_set:Nn \l_document_structure_section_level_int {0}
4664   }
4665   {chapter}{
4666     \int_set:Nn \l_document_structure_section_level_int {1}
4667   }
4668 }{
4669   \str_case:VnF \c_document_structure_class_str {
4670     {book}{
4671       \int_set:Nn \l_document_structure_section_level_int {0}
4672     }
4673     {report}{
4674       \int_set:Nn \l_document_structure_section_level_int {0}
4675     }
4676   }{
4677     \int_set:Nn \l_document_structure_section_level_int {2}
4678   }
4679 }

```

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

```
4680 \def\current@section@level{document}%
4681 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4682 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4683 \cs_new_protected:Npn \skipomgroup {
4684   \ifcase\l_document_structure_section_level_int
4685   \or\stepcounter{part}
4686   \or\stepcounter{chapter}
4687   \or\stepcounter{section}
4688   \or\stepcounter{subsection}
4689   \or\stepcounter{subsubsection}
4690   \or\stepcounter{paragraph}
4691   \or\stepcounter{subparagraph}
4692   \fi
4693 }
```

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

`blindomgroup`

```
4694 \newcommand\at@begin@blindomgroup[1]{%
4695 \newenvironment{blindomgroup}
4696 {
4697   \int_incr:N\l_document_structure_section_level_int
4698   \at@begin@blindomgroup\l_document_structure_section_level_int
4699 }{}}
```

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

```
4700 \newcommand\omgroup@nonum[2]{
4701   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4702   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4703 }
```

(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.

```
4704 \newcommand\omgroup@num[2]{
```

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

```

4705 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4706   \@nameuse{#1}{#2}
4707 }{
4708   \cs_if_exist:NTF\rdfmata@sectioning{
4709     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4710   }{
4711     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4712   }
4713 }
4714 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4715 }

```

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

omgroup

```

4716 \keys_define:nn { document-structure / omgroupp }{
4717   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4718   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4719   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4720   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4721   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4722   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4723   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4724   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4725   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4726   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4727 }
4728 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4729   \str_clear:N \l__document_structure_omgroup_id_str
4730   \str_clear:N \l__document_structure_omgroup_date_str
4731   \clist_clear:N \l__document_structure_omgroup_creators_clist
4732   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4733   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4734   \tl_clear:N \l__document_structure_omgroup_type_tl
4735   \tl_clear:N \l__document_structure_omgroup_short_tl
4736   \tl_clear:N \l__document_structure_omgroup_display_tl
4737   \tl_clear:N \l__document_structure_omgroup_intro_tl
4738   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4739   \keys_set:nn { document-structure / omgroupp } { #1 }
4740 }

```

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

```

4741 \newif\if@mainmatter\@mainmattertrue
4742 \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.

```

4743 \keys_define:nn { document-structure / sectioning }{
4744   name .str_set_x:N = \l__document_structure_sect_name_str ,
4745   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4746   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4747   num .bool_set:N = \l__document_structure_sect_num_bool ,
4748 }

```



```

4749 \cs_new_protected:Nn \__document_structure_sect_args:n {
4750   \str_clear:N \l__document_structure_sect_name_str
4751   \str_clear:N \l__document_structure_sect_ref_str
4752   \bool_set_false:N \l__document_structure_sect_clear_bool
4753   \bool_set_false:N \l__document_structure_sect_num_bool
4754   \keys_set:nn { document-structure / sectioning } { #1 }
4755 }
4756 \newcommand\omdoc@sectioning[3][]{
4757   \__document_structure_sect_args:n {#1}
4758   \let\omdoc@sect@name\l__document_structure_sect_name_str
4759   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4760   \if@mainmatter% numbering not overridden by frontmatter, etc.
4761     \bool_if:NTF \l__document_structure_sect_num_bool {
4762       \omgroup@num{#2}{#3}
4763     }{
4764       \omgroup@nonum{#2}{#3}
4765     }
4766     \def\current@section@level{\omdoc@sect@name}
4767   \else
4768     \omgroup@nonum{#2}{#3}
4769   \fi
4770 }% 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.

```

4771 \newcommand\omgroup@redefine@addtocontents[1]{%
4772   %\edef\__document_structureimport{#1}%
4773   %\@for\@I:=\__document_structureimport\do{%
4774     %\edef\@path{\csname module@\@I @path\endcsname}%
4775     %\@ifundefined{tf@toc}\relax%
4776     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4777   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4778   %\def\addcontentsline##1##2##3{%
4779     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4780   %\else% hyperref.sty not loaded
4781   %\def\addcontentsline##1##2##3{%
4782     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4783   %\fi
4784 }% 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.

```

4785 \int_new:N \l_document_structure_omgroup_level_int
4786 \newenvironment{omgroup}[2][]{% keys, title
4787 {
4788   \__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.

```

4789 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4790   \omgroup@redefine@addtocontents{
4791     %\@ifundefined{module@id}\used@modules%
4792     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4793     }
4794 }

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

4795 \int_incr:N \l_document_structure_omgroup_level_int
4796 \int_incr:N \l_document_structure_section_level_int
4797 \ifcase\l_document_structure_section_level_int
4798   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4799   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4800   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4801   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4802   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4803   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4804   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4805 \fi
4806 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4807 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4808 }% for customization
4809 {}

```

and finally, we localize the sections

```

4810 \newcommand\omdoc@part@kw{Part}
4811 \newcommand\omdoc@chapter@kw{Chapter}
4812 \newcommand\omdoc@section@kw{Section}
4813 \newcommand\omdoc@subsection@kw{Subsection}
4814 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4815 \newcommand\omdoc@paragraph@kw{paragraph}
4816 \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`

```

4817 \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).

```

4818 \cs_if_exist:NTF\frontmatter{
4819   \let\__document_structure_orig_frontmatter\frontmatter
4820   \let\frontmatter\relax
4821 }{
4822   \tl_set:Nn\__document_structure_orig_frontmatter{
4823     \clearpage
4824     \@mainmatterfalse
4825     \pagenumbering{roman}
4826   }
4827 }
4828 \cs_if_exist:NTF\backmatter{

```

```

4829 \let\__document_structure_orig_backmatter\backmatter
4830 \let\backmatter\relax
4831 }{
4832 \tl_set:Nn\__document_structure_orig_backmatter{
4833 \clearpage
4834 \@mainmatterfalse
4835 \pagenumbering{roman}
4836 }
4837 }

```

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.

```

4838 \newenvironment{frontmatter}{
4839 \__document_structure_orig_frontmatter
4840 }{
4841 \cs_if_exist:NTF\mainmatter{
4842 \mainmatter
4843 }{
4844 \clearpage
4845 \@mainmattertrue
4846 \pagenumbering{arabic}
4847 }
4848 }

```

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

```

4849 \newenvironment{backmatter}{
4850 \__document_structure_orig_backmatter
4851 }{
4852 \cs_if_exist:NTF\mainmatter{
4853 \mainmatter
4854 }{
4855 \clearpage
4856 \@mainmattertrue
4857 \pagenumbering{arabic}
4858 }
4859 }

```

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

```

4860 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4861 \def \c__document_structure_document_str{document}
4862 \newcommand\afterprematurestop{}
4863 \def\prematurestop@endomgroup{
4864 \unless\ifx\@currenvir\c__document_structure_document_str
4865 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4866 \expandafter\prematurestop@endomgroup
4867 \fi
4868 }
4869 \providecommand\prematurestop{

```

```

4870 \message{Stopping~sTeX~processing~prematurely}
4871 \prematurestop@endomgroup
4872 \afterprematurestop
4873 \end{document}
4874 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

4875 \RequirePackage{etoolbox}
4876 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4877 \newrobustcmd\useSGvar[1]{%
4878 \@ifundefined{sTeX@Gvar@#1}
4879 {\PackageError{omdoc}
4880 {The sTeX Global variable #1 is undefined}
4881 {set it with \protect\setSGvar}}
4882 \@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.

```

4883 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4884 \@ifundefined{sTeX@Gvar@#1}
4885 {\PackageError{omdoc}
4886 {The sTeX Global variable #1 is undefined}
4887 {set it with \protect\setSGvar}}
4888 {\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.

```
4889 \*cls)
4890 \@@=mikoslides)
4891 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4892 \RequirePackage{l3keys2e,expl-keystr-compatible}
4893
4894 \keys_define:nn{mikoslides / cls}{
4895   class .code:n = {
4896     \PassOptionsToClass{\CurrentOption}{omdoc}
4897     \str_if_eq:nnT{#1}{book}{
4898       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4899     }
4900     \str_if_eq:nnT{#1}{report}{
4901       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4902     }
4903   },
4904   notes .bool_set:N = \c__mikoslides_notes_bool ,
4905   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4906   unknown .code:n = {
4907     \PassOptionsToClass{\CurrentOption}{omdoc}
4908     \PassOptionsToClass{\CurrentOption}{beamer}
4909     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4910   }
4911 }
4912 \ProcessKeysOptions{ mikoslides / cls }
4913 \bool_if:NTF \c__mikoslides_notes_bool {
4914   \PassOptionsToPackage{notes=true}{mikoslides}
4915 }{
4916   \PassOptionsToPackage{notes=false}{mikoslides}
4917 }
4918 \</cls)
```

now we do the same for the mikoslides package.

```

4919 <*package>
4920 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4921 \RequirePackage{l3keys2e,expl-keystr-compat}
4922
4923 \keys_define:nn{mikoslides / pkg}{
4924   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4925   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4926   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4927   slides        .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4928   sectocframes  .bool_set:N = \c__mikoslides_sectocframes_bool ,
4929   frameimages   .bool_set:N = \c__mikoslides_frameimages_bool ,
4930   fiboxed       .bool_set:N = \c__mikoslides_fiboxed_bool ,
4931   nopproblems   .bool_set:N = \c__mikoslides_nopproblems_bool,
4932   unknown       .code:n      = {
4933     \PassOptionsToClass{\CurrentOption}{stex}
4934     \PassOptionsToClass{\CurrentOption}{tikzinput}
4935   }
4936 }
4937 \ProcessKeysOptions{ mikoslides / pkg }
4938 \newif\ifnotes
4939 \bool_if:NTF \c__mikoslides_notes_bool {
4940   \notesttrue
4941 }{
4942   \notesfalse
4943 }
4944

```

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

```

4945 \str_if_empty:NTF \c__mikoslides_topsect_str {
4946   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4947 }{
4948   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4949 }
4950 </package>

```

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

```

4951 <*cls>
4952 \bool_if:NTF \c__mikoslides_notes_bool {
4953   \LoadClass{omdoc}
4954 }{
4955   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4956   \newcounter{Item}
4957   \newcounter{paragraph}
4958   \newcounter{subparagraph}
4959   \newcounter{Hfootnote}
4960   \RequirePackage{omdoc}
4961 }

```

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

```

4962 \RequirePackage{mikoslides}
4963 </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.

```

4964 \*package>
4965 \bool_if:NT \c__mikoslides_notes_bool {
4966   \RequirePackage{a4wide}
4967   \RequirePackage{marginnote}
4968   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4969   \RequirePackage{mdframed}
4970   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4971   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4972 }
4973 \RequirePackage{stex-compatibility}
4974 \RequirePackage{stex-tikzinput}
4975 \RequirePackage{etoolbox}
4976 \RequirePackage{amssymb}
4977 \RequirePackage{amsmath}
4978 \RequirePackage{comment}
4979 \RequirePackage{textcomp}
4980 \RequirePackage{url}
4981 \RequirePackage{graphicx}
4982 \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

```

4983 \bool_if:NT \c__mikoslides_notes_bool {
4984   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4985 }

```

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

```

4986 \newcounter{slide}
4987 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4988 \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.

```

4989 \bool_if:NTF \c__mikoslides_notes_bool {
4990   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4991 }{
4992   \excludecomment{note}
4993 }

```

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.

```
4994 \bool_if:NT \c__mikoslides_notes_bool {
4995   \newlength{\slideframewidth}
4996   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4997 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4998   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4999     \bool_set_true:N #1
5000   }{
5001     \bool_set_false:N #1
5002   }
5003 }
5004 \keys_define:nn{mikoslides / frame}{
5005   label .str_set_x:N = \l__mikoslides_frame_label_str,
5006   allowframebreaks .code:n = {
5007     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
5008   },
5009   allowdisplaybreaks .code:n = {
5010     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
5011   },
5012   fragile .code:n = {
5013     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
5014   },
5015   shrink .code:n = {
5016     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
5017   },
5018   squeeze .code:n = {
5019     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
5020   },
5021   t .code:n = {
5022     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
5023   },
5024 }
5025 \cs_new_protected:Nn \__mikoslides_frame_args:n {
5026   \str_clear:N \l__mikoslides_frame_label_str
5027   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
5028   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
5029   \bool_set_true:N \l__mikoslides_frame_fragile_bool
5030   \bool_set_true:N \l__mikoslides_frame_shrink_bool
5031   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
5032   \bool_set_true:N \l__mikoslides_frame_t_bool
5033   \keys_set:nn { mikoslides / frame }{ #1 }
5034 }
```

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

```
5035 \renewenvironment{frame}[1][]{
5036   \__mikoslides_frame_args:n{#1}
5037   \sffamily
5038   \stepcounter{slide}
5039   \def\@currentlabel{\theslide}
5040   \str_if_empty:NF \l__mikoslides_frame_label_str {
5041     \label{\l__mikoslides_frame_label_str}
```


5042 }

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

5043 \def\itemize@level{outer}
5044 \def\itemize@outer{outer}
5045 \def\itemize@inner{inner}
5046 \renewcommand\newpage{\addtocounter{framenum}{1}}
5047 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5048 \renewenvironment{itemize}{
5049 \ifx\itemize@level\itemize@outer
5050 \def\itemize@label{\$\rhd\$}
5051 \fi
5052 \ifx\itemize@level\itemize@inner
5053 \def\itemize@label{\$\scriptstyle\rhd\$}
5054 \fi
5055 \begin{list}
5056 {\itemize@label}
5057 {\setlength{\labelsep}{.3em}
5058 \setlength{\labelwidth}{.5em}
5059 \setlength{\leftmargin}{1.5em}
5060 }
5061 \edef\itemize@level{\itemize@inner}
5062 }{
5063 \end{list}
5064 }

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

5065 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5066 }{
5067 \medskip\miko@slidelabel\end{mdframed}
5068 }

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

\frametitle

5069 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5070 }

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

EdN:21

\pause 21

5071 \bool_if:NT \c__mikoslides_notes_bool {
5072 \newcommand\pause{
5073 }

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

nomtext

5074 \bool_if:NTF \c__mikoslides_notes_bool {
5075 \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5076 }{
5077 \excludecomment{nomtext}
5078 }

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

nomgroup

```
5079 \bool_if:NTF \c__mikoslides_notes_bool {  
5080   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
5081 }{  
5082   \excludecomment{nomgroup}  
5083 }
```

ndefinition

```
5084 \bool_if:NTF \c__mikoslides_notes_bool {  
5085   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}  
5086 }{  
5087   \excludecomment{ndefinition}  
5088 }
```

nassertion

```
5089 \bool_if:NTF \c__mikoslides_notes_bool {  
5090   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}  
5091 }{  
5092   \excludecomment{nassertion}  
5093 }
```

nsproof

```
5094 \bool_if:NTF \c__mikoslides_notes_bool {  
5095   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
5096 }{  
5097   \excludecomment{nproof}  
5098 }
```

nexample

```
5099 \bool_if:NTF \c__mikoslides_notes_bool {  
5100   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}  
5101 }{  
5102   \excludecomment{nexample}  
5103 }
```

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

```
5104 \def\inputref@preskip{\smallskip}  
5105 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5106 \let\orig@inputref\inputref  
5107 \def\inputref{\@ifstar\ninputref\orig@inputref}  
5108 \newcommand\ninputref[2] [] {  
5109   \bool_if:NT \c__mikoslides_notes_bool {  
5110     \orig@inputref[#1]{#2}  
5111   }  
5112 }
```

(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>}`.

```

5113 \newlength{\slidelogoheight}
5114
5115 \bool_if:NTF \c__mikoslides_notes_bool {
5116   \setlength{\slidelogoheight}{.4cm}
5117 }{
5118   \setlength{\slidelogoheight}{1cm}
5119 }
5120 \newsavebox{\slidelogo}
5121 \sbox{\slidelogo}{\TeX}
5122 \newrobustcmd{\setslidelogo}[1]{
5123   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5124 }
```

(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.

```

5125 \def\source{Michael Kohlhase}% customize locally
5126 \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.

```

5127 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5128 \newsavebox{\cclogo}
5129 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5130 \newif\ifcchref\cchreffalse
5131 \AtBeginDocument{
5132   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5133 }
5134 \def\licensing{
5135   \ifcchref
5136     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5137   \else
5138     {\usebox{\cclogo}}
5139   \fi
5140 }
5141 \newrobustcmd{\setlicensing}[2][]{
5142   \def\@url{#1}
5143   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5144   \ifx\@url\@empty
5145     \def\licensing{{\usebox{\cclogo}}}
5146   \else
5147     \def\licensing{
```

```

5148     \ifcchref
5149     \href{#1}{\usebox{\cclogo}}
5150   \else
5151     {\usebox{\cclogo}}
5152   \fi
5153 }
5154 \fi
5155 }

```

(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

```

5156 \newrobustcmd\miko@slidelabel{
5157   \vbox to \slidelogoheight{
5158     \vss\hbox to \slidewidth
5159     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5160   }
5161 }

```

(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.

```

5162 \def\Gin@mhrepos{}
5163 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5164 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5165 \newrobustcmd\frameimage[2][{}{
5166   \stepcounter{slide}
5167   \bool_if:NT \c__mikoslides_frameimages_bool {
5168     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5169     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5170     \begin{center}
5171       \bool_if:NTF \c__mikoslides_fiboxed_bool {
5172         \fbox{
5173           \ifx\Gin@ewidth\@empty
5174             \ifx\Gin@mhrepos\@empty
5175               \mhgraphics[width=\slidewidth,#1]{#2}
5176             \else
5177               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5178             \fi
5179           \else% \Gin@ewidth empty
5180             \ifx\Gin@mhrepos\@empty
5181               \mhgraphics[#1]{#2}
5182             \else
5183               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5184             \fi
5185           \fi% \Gin@ewidth empty
5186         }
5187       }{
5188         \ifx\Gin@ewidth\@empty

```

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

```

5189         \ifx\Gin@mhrepos\@empty
5190             \mhgraphics[width=\slidewidth,#1]{#2}
5191         \else
5192             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5193         \fi
5194         \ifx\Gin@mhrepos\@empty
5195             \mhgraphics[#1]{#2}
5196         \else
5197             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5198         \fi
5199     \fi% Gin@ewidth empty
5200 }
5201 \end{center}
5202 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5203 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5204 }
5205 } % 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.

```

5206 \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

```

5207 \AddToHook{begindocument}{
5208     \definecolor{green}{rgb}{0,.5,0}
5209     \definecolor{purple}{cmyk}{.3,1,0,.17}
5210 }

```

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`.

```

5211 % \def\STpresent#1{\textcolor{blue}{#1}}
5212 \def\defemph#1{\textcolor{magenta}{#1}}
5213 \def\symrefemph#1{\textcolor{cyan}{#1}}
5214 \def\compemph#1{\textcolor{blue}{#1}}
5215 \def\titleemph#1{\textcolor{blue}{#1}}
5216 \def\__omtext_lec#1(\textcolor{green}{#1})

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

5217 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5218 \def\smalltextwarning{
5219     \pgfuseimage{miko@small@dbend}
5220     \xspace
5221 }
5222 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5223 \newrobustcmd\textwarning{
5224   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5225   \xspace
5226 }
5227 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5228 \newrobustcmd\bigtextwarning{
5229   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5230   \xspace
5231 }

```

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

```

5232 \newrobustcmd\putgraphicsat[3]{
5233   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5234 }
5235 \newrobustcmd\putat[2]{
5236   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5237 }

```

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`.

```

5238 \bool_if:NT \c__mikoslides_sectocframes_bool {
5239   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5240     \newcounter{chapter}\counterwithin*{section}{chapter}
5241   }{
5242     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5243       \newcounter{chapter}\counterwithin*{section}{chapter}
5244     }
5245   }
5246 }

```

`\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
5247 \def\part@prefix{}
5248 \@ifpackageloaded{omdoc}{}{
5249   \str_case:VnF \__mikoslidestopsect {
5250     {part}{
5251       \int_set:Nn \l_document_structure_section_level_int {0}
5252       \def\thesection{\arabic{chapter}.\arabic{section}}
5253       \def\part@prefix{\arabic{chapter}.}
5254     }
5255     {chapter}{
5256       \int_set:Nn \l_document_structure_section_level_int {1}
5257       \def\thesection{\arabic{chapter}.\arabic{section}}
5258       \def\part@prefix{\arabic{chapter}.}
5259     }
5260   }{
5261     \int_set:Nn \l_document_structure_section_level_int {2}
5262     \def\part@prefix{}

```

```

5263 }
5264 }
5265
5266 \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

```

5267 \renewenvironment{omgroup}[2][]{
5268   \__document_structure_omgroup_args:n { #1 }
5269   \int_incr:N \l_document_structure_omgroup_level_int
5270   \int_incr:N \l_document_structure_section_level_int
5271   \bool_if:NT \c__mikoslides_sectocframes_bool {
5272     \stepcounter{slide}
5273     \begin{frame}[noframenumbering]
5274     \vfill\Large\centering
5275     \red{
5276       \ifcase\l_document_structure_section_level_int\or
5277         \stepcounter{part}
5278         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5279         \def\currentsectionlevel{\omdoc@part@kw}
5280       \or
5281         \stepcounter{chapter}
5282         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5283         \def\currentsectionlevel{\omdoc@chapter@kw}
5284       \or
5285         \stepcounter{section}
5286         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5287         \def\currentsectionlevel{\omdoc@section@kw}
5288       \or
5289         \stepcounter{subsection}
5290         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5291         \def\currentsectionlevel{\omdoc@subsection@kw}
5292       \or
5293         \stepcounter{subsubsection}
5294         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5295         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5296       \or
5297         \stepcounter{paragraph}
5298         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5299         \def\currentsectionlevel{\omdoc@paragraph@kw}
5300       \else
5301         \def\__mikoslideslabel{}
5302         \def\currentsectionlevel{\omdoc@paragraph@kw}
5303       \fi% end ifcase
5304       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5305       \quad #2%
5306     }%
5307     \vfill%
5308     \end{frame}%
5309   }
5310   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5311 }{}
5312 }

```

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

```

5313 \def\inserttheorembodyfont{\normalfont}
5314 %\bool_if:NF \c__mikoslides_notes_bool {
5315 % \defbeamertemplate{theorem begin}{miko}
5316 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5317 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5318 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5319 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5320 % \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.

```

5321 % \expandafter\def\csname Parent2\endcsname{}
5322 %}
5323
5324 \AddToHook{begindocument}{ % this does not work for some reasons
5325 \setbeamertemplate{theorems}[ams style]
5326 }
5327 \bool_if:NT \c__mikoslides_notes_bool {
5328 \renewenvironment{columns}[1][{}]{%
5329 \par\noindent%
5330 \begin{minipage}%
5331 \slidewidth\centering\leavevmode%
5332 }{%
5333 \end{minipage}\par\noindent%
5334 }%
5335 \newsavebox\columnbox%
5336 \renewenvironment<>{column}[2][{}]{%
5337 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5338 }{%
5339 \end{minipage}\end{lrbox}\usebox\columnbox%
5340 }%
5341 }
5342 \bool_if:NTF \c__mikoslides_noproblems_bool {
5343 \newenvironment{problems}{}{}
5344 }{
5345 \excludecomment{problems}
5346 }

```

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.

```

5347 \gdef\printexcursions{}
5348 \newcommand\excursionref[2]{% label, text
5349 \bool_if:NT \c__mikoslides_notes_bool {

```



```

5350 \begin{sparagraph}[title=Excursion]
5351 #2 \sref[fallback=the appendix]{#1}.
5352 \end{sparagraph}
5353 }
5354 }
5355 \newcommand\activate@excursion[2][]{
5356 \gappto\printexcursions{\inputref[#1]{#2}}
5357 }
5358 \newcommand\excursion[4][]{% repos, label, path, text
5359 \bool_if:NT \c__mikoslides_notes_bool {
5360 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5361 }
5362 }

```

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

\excursiongroup

```

5363 \keys_define:nn{mikoslides / excursiongroup }{
5364 id .str_set_x:N = \l__mikoslides_excursion_id_str,
5365 intro .tl_set:N = \l__mikoslides_excursion_intro_tl,
5366 mhrepos .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5367 }
5368 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5369 \tl_clear:N \l__mikoslides_excursion_intro_tl
5370 \str_clear:N \l__mikoslides_excursion_id_str
5371 \str_clear:N \l__mikoslides_excursion_mhrepos_str
5372 \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5373 }
5374 \newcommand\excursiongroup[1][]{
5375 \__mikoslides_excursion_args:n{ #1 }
5376 \ifdefempty\printexcursions{}% only if there are excursions
5377 {\begin{note}
5378 \begin{omgroup}[#1]{Excursions}%
5379 \ifdefempty\l__mikoslides_excursion_intro_tl{{
5380 \inputref[\l__mikoslides_excursion_mhrepos_str]{
5381 \l__mikoslides_excursion_intro_tl
5382 }
5383 }
5384 \printexcursions%
5385 \end{omgroup}
5386 \end{note}}
5387 }
5388 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5389 \end{package}

```

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

Chapter 40

The Implementation

40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
5390 <*package>
5391 <@@=problems>
5392 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5393 \RequirePackage{l3keys2e,expl-keystr-compatible}
5394
5395 \keys_define:nn { problem / pkg }{
5396   notes      .default:n    = { true },
5397   notes      .bool_set:N   = \c__problems_notes_bool,
5398   gnotes     .default:n    = { true },
5399   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5400   hints      .default:n    = { true },
5401   hints      .bool_set:N   = \c__problems_hints_bool,
5402   solutions  .default:n    = { true },
5403   solutions  .bool_set:N   = \c__problems_solutions_bool,
5404   pts        .default:n    = { true },
5405   pts        .bool_set:N   = \c__problems_pts_bool,
5406   min        .default:n    = { true },
5407   min        .bool_set:N   = \c__problems_min_bool,
5408   boxed      .default:n    = { true },
5409   boxed      .bool_set:N   = \c__problems_boxed_bool,
5410   unknown    .code:n       = {}
5411 }
5412 \def\solutionstrue{
5413   \bool_set_true:N \c__problems_solutions_bool
5414 }
5415 \def\solutionsfalse{
5416   \bool_set_false:N \c__problems_solutions_bool
5417 }
5418
5419 \ProcessKeysOptions{ problem / pkg }
```

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

```

5420 \RequirePackage{stex-compatibility}
5421 \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.

```

5422 \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.

```

5423 \def\prob@problem@kw{Problem}
5424 \def\prob@solution@kw{Solution}
5425 \def\prob@hint@kw{Hint}
5426 \def\prob@note@kw{Note}
5427 \def\prob@gnote@kw{Grading}
5428 \def\prob@pt@kw{pt}
5429 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5430 \AddToHook{begindocument}{
5431   \ltx@ifpackageloaded{babel}{
5432     \makeatletter
5433     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5434     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5435       \input{problem-ngerman.ldf}
5436     }
5437     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5438       \input{problem-finnish.ldf}
5439     }
5440     \clist_if_in:NnT \l_tmpa_clist {french}{
5441       \input{problem-french.ldf}
5442     }
5443     \clist_if_in:NnT \l_tmpa_clist {russian}{
5444       \input{problem-russian.ldf}
5445     }
5446     \makeatother
5447   }{}
5448 }

```

40.2 Problems and Solutions

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

```

5449 \keys_define:nn{ problem / problem }{
5450   id      .str_set:x:N = \l__problems_prob_id_str,
5451   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5452   min     .tl_set:N    = \l__problems_prob_min_tl,
5453   title   .tl_set:N    = \l__problems_prob_title_tl,
5454   refnum  .int_set:N   = \l__problems_prob_refnum_int
5455 }
5456 \cs_new_protected:Nn \__problems_prob_args:n {

```

```

5457 \str_clear:N \l__problems_prob_id_str
5458 \tl_clear:N \l__problems_prob_pts_tl
5459 \tl_clear:N \l__problems_prob_min_tl
5460 \tl_clear:N \l__problems_prob_title_tl
5461 \int_zero_new:N \l__problems_prob_refnum_int
5462 \keys_set:nn { problem / problem }{ #1 }
5463 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5464   \let\l__problems_inclprob_refnum_int\undefined
5465 }
5466 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5467 \newcounter{problem}
5468 \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.

```

5469 \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

```

5470 \newcommand\prob@number{
5471   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5472     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5473   }{
5474     \int_if_exist:NTF \l__problems_prob_refnum_int {
5475       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5476     }{
5477       \prob@label\theproblem
5478     }
5479   }
5480 }

```

(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.

```

5481 \newcommand\prob@title[3]{%
5482   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5483     #2 \l__problems_inclprob_title_tl #3
5484   }{
5485     \tl_if_exist:NTF \l__problems_prob_title_tl {
5486       #2 \l__problems_prob_title_tl #3
5487     }{
5488       #1
5489     }
5490   }
5491 }

```

(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.

```

5492 \def\prob@heading{
5493   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5494   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5495 }

```

(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`

```

5496 \newenvironment{problem}[1][1]{
5497   \_problems_prob_args:n{#1}%\sref@target%
5498   \@in@omtexttrue% we are in a statement (for inline definitions)
5499   \stepcounter{problem}\record@problem
5500   \def\current@section@level{\prob@problem@kw}
5501   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5502 }%
5503   {\smallskip}
5504   \bool_if:NT \c__problems_boxed_bool {
5505     \surroundwithmdframed{problem}
5506   }

```

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

```

5507 \def\record@problem{
5508   \protected@write\@auxout{}
5509   {
5510     \string\@problem{\prob@number}
5511     {
5512       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5513         \l__problems_inclprob_pts_tl
5514       }{
5515         \l__problems_prob_pts_tl
5516       }
5517     }%
5518     {
5519       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5520         \l__problems_inclprob_min_tl
5521       }{
5522         \l__problems_prob_min_tl
5523       }
5524     }
5525   }
5526 }

```

(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).

```

5527 \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.

```

5528 \keys_define:nn { problem / solution }{
5529   id          .str_set_x:N = \l__problems_solution_id_str ,
5530   for         .tl_set:N    = \l__problems_solution_for_tl ,
5531   height      .dim_set:N   = \l__problems_solution_height_dim ,
5532   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5533   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5534   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5535 }
5536 \cs_new_protected:Nn \__problems_solution_args:n {
5537   \str_clear:N \l__problems_solution_id_str
5538   \tl_clear:N \l__problems_solution_for_tl
5539   \tl_clear:N \l__problems_solution_srccite_tl
5540   \clist_clear:N \l__problems_solution_creators_clist
5541   \clist_clear:N \l__problems_solution_contributors_clist
5542   \dim_zero:N \l__problems_solution_height_dim
5543   \keys_set:nn { problem / solution }{ #1 }
5544 }

```

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

```

5545 \newcommand\@startsolution[1][ ]{
5546   \__problems_solution_args:n { #1 }
5547   \@in@omtexttrue% we are in a statement.
5548   \bool_if:NF \c__problems_boxed_bool { \hrule }
5549   \smallskip\noindent
5550   {\textbf\prob@solution@kw : \enspace}
5551   \begin{small}
5552   \def\current@section@level{\prob@solution@kw}
5553   \ignorespacesandpars
5554 }

```

\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.

```

5555 \newcommand\startsolutions{
5556   \specialcomment{solution}{\@startsolution}{
5557     \bool_if:NF \c__problems_boxed_bool {
5558       \hrule\medskip
5559     }
5560     \end{small}%
5561   }
5562   \bool_if:NT \c__problems_boxed_bool {
5563     \surroundwithmdframed{solution}
5564   }
5565 }

```

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

\stopsolutions

```

5566 \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.

```

5567 \bool_if:NTF \c_problems_solutions_bool {
5568   \startsolutions
5569 }{
5570   \stopsolutions
5571 }

```

exnote

```

5572 \bool_if:NTF \c_problems_notes_bool {
5573   \newenvironment{exnote}[1][{}{
5574     \par\smallskip\hrule\smallskip
5575     \noindent\textbf{\prob@note@kw : }\small
5576   }{
5577     \smallskip\hrule
5578   }
5579 }{
5580   \excludecomment{exnote}
5581 }

```

hint

```

5582 \bool_if:NTF \c_problems_notes_bool {
5583   \newenvironment{hint}[1][{}{
5584     \par\smallskip\hrule\smallskip
5585     \noindent\textbf{\prob@hint@kw :~ }\small
5586   }{
5587     \smallskip\hrule
5588   }
5589   \newenvironment{exhint}[1][{}{
5590     \par\smallskip\hrule\smallskip
5591     \noindent\textbf{\prob@hint@kw :~ }\small
5592   }{
5593     \smallskip\hrule
5594   }
5595 }{
5596   \excludecomment{hint}
5597   \excludecomment{exhint}
5598 }

```

gnote

```

5599 \bool_if:NTF \c_problems_notes_bool {
5600   \newenvironment{gnote}[1][{}{
5601     \par\smallskip\hrule\smallskip
5602     \noindent\textbf{\prob@gnote@kw : }\small
5603   }{
5604     \smallskip\hrule
5605   }
5606 }{
5607   \excludecomment{gnote}
5608 }

```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
5609 \newenvironment{mcb}{
5610   \begin{enumerate}
5611 }{
5612   \end{enumerate}
5613 }
```

we define the keys for the mcc macro

```
5621 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5622   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5623     \bool_set_true:N #1
5624   }{
5625     \bool_set_false:N #1
5626   }
5627 }
5628 }
5629 \keys_define:nn { problem / mcc }{
5630   id          .str_set:x:N = \l__problems_mcc_id_str ,
5631   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5632   T           .default:n    = { true } ,
5633   T           .bool_set:N    = \l__problems_mcc_t_bool ,
5634   F           .default:n    = { true } ,
5635   F           .bool_set:N    = \l__problems_mcc_f_bool ,
5636   Ttext       .code:n       = {
5637     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5638   } ,
5639   Ftext       .code:n       = {
5640     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5641   }
5642 }
5643 }
5644 \cs_new_protected:Nn \l__problems_mcc_args:n {
5645   \str_clear:N \l__problems_mcc_id_str
5646   \tl_clear:N \l__problems_mcc_feedback_tl
5647   \bool_set_true:N \l__problems_mcc_t_bool
5648   \bool_set_true:N \l__problems_mcc_f_bool
5649   \bool_set_true:N \l__problems_mcc_Ttext_bool
5650   \bool_set_false:N \l__problems_mcc_Ftext_bool
5651   \keys_set:nn { problem / mcc }{ #1 }
5652 }
```

\mcc

```
5644 \newcommand\mcc[2][]{
5645   \l__problems_mcc_args:n{ #1 }
5646   \item #2
5647   \bool_if:NT \c__problems_solutions_bool {
5648     \
5649     \bool_if:NT \l__problems_mcc_t_bool {
5650       % TODO!
5651       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5652     }
5653     \bool_if:NT \l__problems_mcc_f_bool {
```

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


```

5654      % TODO!
5655      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5656    }
5657    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5658      !
5659    }{
5660      \l__problems_mcc_feedback_tl
5661    }
5662  }
5663 } %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.

```

5664
5665 \keys_define:nn{ problem / inclproblem }{
5666   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5667   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5668   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5669   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5670   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5671   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5672 }
5673 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5674   % \str_clear:N \l__problems_prob_id_str
5675   \tl_clear:N \l__problems_inclprob_pts_tl
5676   \tl_clear:N \l__problems_inclprob_min_tl
5677   \tl_clear:N \l__problems_inclprob_title_tl
5678   \int_zero_new:N \l__problems_inclprob_refnum_int
5679   \str_clear:N \l__problems_inclprob_mhrepos_str
5680   \keys_set:nn { problem / inclproblem }{ #1 }
5681   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5682     \let\l__problems_inclprob_pts_tl\undefined
5683   }
5684   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5685     \let\l__problems_inclprob_min_tl\undefined
5686   }
5687   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5688     \let\l__problems_inclprob_title_tl\undefined
5689   }
5690   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5691     \let\l__problems_inclprob_refnum_int\undefined
5692   }
5693 }
5694
5695 \cs_new_protected:Nn \l__problems_inclprob_clear: {
5696   % \str_clear:N \l__problems_prob_id_str
5697   \let\l__problems_inclprob_pts_tl\undefined
5698   \let\l__problems_inclprob_min_tl\undefined

```

```

5699 \let\l__problems_inclprob_title_tl\undefined
5700 \let\l__problems_inclprob_refnum_int\undefined
5701 \let\l__problems_inclprob_mhrepos_str\undefined
5702 }
5703
5704 \newcommand\includeproblem[2][ ]{
5705   \__problems_inclprob_args:n{ #1 }
5706   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5707     \input{#2}
5708   }{
5709     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5710       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5711     }
5712   }
5713   \__problems_inclprob_clear:
5714 }

```

(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.

```

5715 \AddToHook{enddocument}{
5716   \bool_if:NT \c__problems_pts_bool {
5717     \message{Total:~\arabic{pts}~points}
5718   }
5719   \bool_if:NT \c__problems_min_bool {
5720     \message{Total:~\arabic{min}~minutes}
5721   }
5722 }

```

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

```

5723 \def\pts#1{
5724   \bool_if:NT \c__problems_pts_bool {
5725     \marginpar{#1~\prob@pt@kw}
5726   }
5727 }
5728 \def\min#1{
5729   \bool_if:NT \c__problems_min_bool {
5730     \marginpar{#1~\prob@min@kw}
5731   }
5732 }

```

\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.

```

5733 \newcounter{pts}
5734 \def\show@pts{
5735   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5736     \bool_if:NT \c__problems_pts_bool {
5737       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5738       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5739     }
5740   }{
5741     \tl_if_exist:NT \l__problems_prob_pts_tl {
5742       \bool_if:NT \c__problems_pts_bool {
5743         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5744         \addtocounter{pts}{\l__problems_prob_pts_tl}
5745       }
5746     }
5747   }
5748 }

```

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

\show@min

```

5749 \newcounter{min}
5750 \def\show@min{
5751   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5752     \bool_if:NT \c__problems_min_bool {
5753       \marginpar{\l__problems_inclprob_pts_tl;min}
5754       \addtocounter{min}{\l__problems_inclprob_min_tl}
5755     }
5756   }{
5757     \tl_if_exist:NT \l__problems_prob_min_tl {
5758       \bool_if:NT \c__problems_min_bool {
5759         \marginpar{\l__problems_prob_min_tl;min}
5760         \addtocounter{min}{\l__problems_prob_min_tl}
5761       }
5762     }
5763   }
5764 }
5765 \</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.

```
5766 <@@=hwexam>
5767 <*cls>
5768 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5769 \RequirePackage{l3keys2e,expl-keystr-compatible}
5770 \DeclareOption*{
5771   \PassOptionsToClass{\CurrentOption}{omdoc}
5772   \PassOptionsToPackage{\CurrentOption}{stex}
5773   \PassOptionsToPackage{\CurrentOption}{hwexam}
5774   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5775 }
5776 \ProcessOptions
```

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

```
5777 \LoadClass{omdoc}
5778 \RequirePackage{stex}
5779 \RequirePackage{hwexam}
5780 \RequirePackage{tikzinput}
5781 \RequirePackage{graphicx}
5782 \RequirePackage{a4wide}
5783 \RequirePackage{amssymb}
5784 \RequirePackage{amstext}
5785 \RequirePackage{amsmath}
```

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

```

5786 \newcommand\assig@default@type{\hwexam@assignment@kw}
5787 \def\document@hwexamtype{\assig@default@type}
5788 <@@=document_structure>
5789 \keys_define:nn { document-structure / document }{
5790 id .str_set_x:N = \c_document_structure_document_id_str,
5791 hwexamtype .tl_set:N = \document@hwexamtype
5792 }
5793 <@@=hwexam>
5794 </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.

```
5795 \langle *package\rangle
5796 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5797 \RequirePackage{l3keys2e,expl-keystr-compat}
5798
5799 \newif\iftest\testfalse
5800 \DeclareOption{test}{\testtrue}
5801 \newif\ifmultiple\multiplefalse
5802 \DeclareOption{multiple}{\multipletrue}
5803 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5804 \ProcessOptions
```

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

```
5805 \RequirePackage{keyval}[1997/11/10]
5806 \RequirePackage{problem}
```

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

```
5807 \newcommand\hwexam@assignment@kw{Assignment}
5808 \newcommand\hwexam@given@kw{Given}
5809 \newcommand\hwexam@due@kw{Due}
5810 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5811 blank~for~extra~space}%
5812 \newcommand\correction@probs@kw{prob.}%
5813 \newcommand\correction@pts@kw{total}%
5814 \newcommand\correction@reached@kw{reached}%
5815 \newcommand\correction@sum@kw{Sum}%
5816 \newcommand\correction@grade@kw{grade}%
5817 \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

```

5818 \AddToHook{begindocument}{
5819 \ltx@ifpackageloaded{babel}{
5820 \makeatletter
5821 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5822 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5823 \input{hwexam-ngerman.ldf}
5824 }
5825 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5826 \input{hwexam-finnish.ldf}
5827 }
5828 \clist_if_in:NnT \l_tmpa_clist {french}{
5829 \input{hwexam-french.ldf}
5830 }
5831 \clist_if_in:NnT \l_tmpa_clist {russian}{
5832 \input{hwexam-russian.ldf}
5833 }
5834 \makeatother
5835 }{}
5836 }
5837

```

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.

```

5838 \newcounter{assignment}
5839 \numberproblemsin{assignment}
5840 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5841 \keys_define:nn { hwexam / assignment } {
5842 id .str_set:N = \l__hwexam_assign_id_str,
5843 number .int_set:N = \l__hwexam_assign_number_int,
5844 title .tl_set:N = \l__hwexam_assign_title_tl,
5845 type .tl_set:N = \l__hwexam_assign_type_tl,
5846 given .tl_set:N = \l__hwexam_assign_given_tl,
5847 due .tl_set:N = \l__hwexam_assign_due_tl,
5848 loadmodules .code:n = {
5849 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5850 }
5851 }
5852 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5853 \str_clear:N \l__hwexam_assign_id_str
5854 \int_set:Nn \l__hwexam_assign_number_int {-1}
5855 \tl_clear:N \l__hwexam_assign_title_tl
5856 \tl_clear:N \l__hwexam_assign_type_tl
5857 \tl_clear:N \l__hwexam_assign_given_tl
5858 \tl_clear:N \l__hwexam_assign_due_tl
5859 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

5860 \keys_set:nn { hwexam / assignment }{ #1 }
5861 }

```

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.

```

5862 \newcommand\given@due[2]{
5863 \bool_lazy_all:nF {
5864 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5865 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5866 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5867 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5868 }{ #1 }
5869
5870 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5871 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5872 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5873 }
5874 }{
5875 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5876 }
5877
5878 \bool_lazy_or:nnF {
5879 \bool_lazy_and_p:nn {
5880 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5881 }{
5882 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5883 }
5884 }{
5885 \bool_lazy_and_p:nn {
5886 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5887 }{
5888 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5889 }
5890 }{ ,~ }
5891
5892 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5893 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5894 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5895 }
5896 }{
5897 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5898 }
5899
5900 \bool_lazy_all:nF {
5901 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5902 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5903 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5904 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5905 }{ #2 }
5906 }

```

`\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.

```

5907 \newcommand\assignment@title[3]{
5908 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5909 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5910 #1
5911 }{
5912 #2\l__hwexam_assign_title_tl#3
5913 }
5914 }{
5915 #2\l__hwexam_inclassassign_title_tl#3
5916 }
5917 }

```

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

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

```

5918 \newcommand\assignment@number{
5919 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5920 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5921 \int_use:N \l__hwexam_assign_number_int
5922 }
5923 }{
5924 \int_use:N \l__hwexam_inclassassign_number_int
5925 }
5926 }

```

(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.

```

5927 \newenvironment{assignment}[1][]{
5928 \__hwexam_assignment_args:n { #1 }
5929 %\sref@target
5930 \let\__hwexamnum\l__hwexam_assign_number_int
5931 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5932 \stepcounter{assignment}
5933 }{
5934 \setcounter{assignment}{\int_use:N\__hwexamnum}
5935 }
5936 \setcounter{problem}{0}
5937 \def\current@section@level{\document@hwexamtype}
5938 %\sref@label@id{\document@hwexamtype \thesection}
5939 \begin{@assignment}
5940 }{
5941 \end{@assignment}
5942 }

```

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

```

5943 \def\__hwexasstitle{
5944 \protect\document@hwexamtype~\arabic{assignment}
5945 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
5946 }
5947 \ifmultiple
5948 \newenvironment{@assignment}{
5949 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5950 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5951 }{
5952 \begin{omgroup}{\__hwexasstitle}
5953 }
5954 }{
5955 \end{omgroup}
5956 }

```

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

```

5957 \else
5958 \newenvironment{@assignment}{
5959 \begin{center}\bf
5960 \Large@title\strut\
5961 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
5962 \large\given@due{--;\}\{;\}--}
5963 \end{center}
5964 }{}
5965 \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.

```

5966 \keys_define:nn { hwexam / inclassignment } {
5967 %id .str_set_x:N = \l__hwexam_assign_id_str,
5968 number .int_set:N = \l__hwexam_inclassign_number_int,
5969 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5970 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5971 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5972 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5973 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5974 }
5975 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5976 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5977 \tl_clear:N \l__hwexam_inclassign_title_tl
5978 \tl_clear:N \l__hwexam_inclassign_type_tl
5979 \tl_clear:N \l__hwexam_inclassign_given_tl
5980 \tl_clear:N \l__hwexam_inclassign_due_tl
5981 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5982 \keys_set:nn { hwexam / inclassignment }{ #1 }
5983 }
5984 \__hwexam_inclassignment_args:n {}
5985
5986 \newcommand\inputassignment[2][{}]{

```

```

5987 \_hwexam_inclassnment_args:n { #1 }
5988 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
5989 \input{#2}
5990 }{
5991 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
5992 \input{\mhp{path}{\l__hwexam_inclassn_mhrepos_str}{#2}}
5993 }
5994 }
5995 \_hwexam_inclassnment_args:n {}
5996 }
5997 \newcommand\includeassignment[2][]{
5998 \newpage
5999 \inputassignment[#1]{#2}
6000 }

```

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

42.4 Typesetting Exams

\quizheading

```

6001 \ExplSyntaxOff
6002 \newcommand\quizheading[1]{%
6003 \def\@tas{#1}%
6004 \large\noindent NAME: \hspace{8cm} MAILBOX:\|[2ex]%
6005 \ifx\@tas\@empty\else%
6006 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\|[2ex]%
6007 \fi%
6008 }
6009 \ExplSyntaxOn

```

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

\testheading

```

6010 \keys_define:nn { hwexam / testheading } {
6011 min .tl_set:N = \l__hwexam_testheading_min_tl,
6012 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
6013 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
6014 }
6015 \cs_new_protected:Nn \l__hwexam_testheading_args:n {
6016 \tl_clear:N \l__hwexam_testheading_min_tl
6017 \tl_clear:N \l__hwexam_testheading_duration_tl
6018 \tl_clear:N \l__hwexam_testheading_reqpts_tl
6019 \keys_set:nn { hwexam / testheading }{ #1 }
6020 }
6021 \newenvironment{testheading}[1][]{
6022 \_hwexam_testheading_args:n{ #1 }
6023 \noindent\large{Name:~\hfill
6024 Matriculation Number:\hspace*{2cm}\strut}\|[1ex]
6025 \begin{center}
6026 \Large\textbf{\@title}\|[1ex]
6027 \large\@date\|[3ex]
6028 \end{center}
6029 \textbf{You~have~

```

```

6030 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
6031 {\l__hwexam_testheading_min_tl}~minutes
6032 }{
6033 {\l__hwexam_testheading_duration_tl}
6034 }~
6035 (sharp)~for~the~test
6036 };\\
6037 Write~the~solutions~to~the~sheet.
6038 \par\noindent
6039 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
6040 \advance\check@time by -\theassignment@totalmin
6041 The~estimated~time~for~solving~this~exam~is~
6042 {\theassignment@totalmin}~minutes,~
6043 leaving~you~{\the\check@time}~minutes~for~revising~
6044 your~exam.
6045
6046 \par\noindent
6047 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
6048 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
6049 You~can~reach~{\theassignment@totalpts}~points~if~you~
6050 solve~all~problems.~You~will~only~need~
6051 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
6052 i.e.\ {\the\bonus@pts}~points~are~bonus~points.
6053 \vfill
6054 \begin{center}
6055 {
6056 \Large\em You~have~ample~time,~so~take~it~slow~
6057 and~avoid~rushing~to~mistakes!\\[2ex]
6058 Different~problems~test~different~skills~and~
6059 knowledge,~so~do~not~get~stuck~on~one~problem.
6060 }
6061 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\\[3ex]
6062 \end{center}
6063 }{
6064 \newpage
6065 }

```

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

\testspace

```

6066 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6067 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6068 \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.

```

6069 <@@=problems>
6070 \renewcommand\@problem[3]{
6071 \stepcounter{assignment@probs}
6072 \def\__problemspts{#2}
6073 \ifx\__problemspts\@empty\else
6074 \addtocounter{assignment@totalpts}{#2}
6075 \fi
6076 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6077 \xdef\correction@probs{\correction@probs & #1}%
6078 \xdef\correction@pts{\correction@pts & #2}
6079 \xdef\correction@reached{\correction@reached &}
6080 }
6081 <@@=hwexam>

```

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

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

```

6082 \newcounter{assignment@probs}
6083 \newcounter{assignment@totalpts}
6084 \newcounter{assignment@totalmin}
6085 \def\correction@probs{\correction@probs@kw}%
6086 \def\correction@pts{\correction@pts@kw}%
6087 \def\correction@reached{\correction@reached@kw}%
6088 \def\after@correction@table{}%
6089 \stepcounter{assignment@probs}
6090 \newcommand\correction@table{
6091 \resizebox{\textwidth}{!}{%
6092 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6093 &\multicolumn{\theassignment@probs}{c|}|%|
6094 {\footnotesize\correction@forgrading@kw} &\\ \hline
6095 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6096 \correction@pts & \theassignment@totalpts & \\ \hline
6097 \correction@reached & & \[.7cm]\hline
6098 \end{tabular}}
6099 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
6100 </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}
```