

The sTeX3 Package *

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

2022-02-08

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 2022-02-08)

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	Primitive Symbols (The sTeX Metatheory)	9
6	sTeX Statements (Definitions, Theorems, Examples, ...)	10
7	Additional Packages	11
7.1	Modular Document Structuring	11
7.2	Slides and Course Notes	11
7.3	Homework, Problems and Exams	11
8	Stuff	12
8.1	Modules	12
8.1.1	Semantic Macros and Notations	12
	Other Argument Types	14
	Precedences	16
8.1.2	Archives and Imports	16
	Namespaces	16
	Paths in Import-Statements	17
II	Documentation	18
9	sTeX -Basics	19
9.1	Macros and Environments	19
10	sTeX -MathHub	21
10.1	Macros and Environments	21
10.1.1	Files, Paths, URIs	21
10.1.2	MathHub Archives	22
11	sTeX -References	24
11.1	Macros and Environments	24

12	sTeX-Modules	25
12.1	Macros and Environments	25
12.1.1	The <code>module</code> -environment	27
13	sTeX-Module Inheritance	30
13.1	Macros and Environments	30
13.1.1	SMS Mode	30
13.1.2	Imports and Inheritance	31
14	sTeX-Symbols	34
14.1	Macros and Environments	34
15	sTeX-Terms	37
15.1	Macros and Environments	37
16	sTeX-Structural Features	40
16.1	Macros and Environments	40
16.1.1	Structures	40
17	sTeX-Statements	41
17.1	Macros and Environments	41
18	sTeX-Proofs: Structural Markup for Proofs	42
18.1	Introduction	44
18.2	The User Interface	45
18.2.1	Package Options	45
18.2.2	Proofs and Proof steps	45
18.2.3	Justifications	45
18.2.4	Proof Structure	46
18.2.5	Proof End Markers	47
18.2.6	Configuration of the Presentation	47
18.3	Limitations	47
19	sTeX-Metatheory	49
19.1	Symbols	49
III	Extensions	50
20	Tikzinput	51
20.1	Macros and Environments	51

21	document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX	52
21.1	Introduction	52
21.2	The User Interface	53
21.2.1	Package and Class Options	53
21.2.2	Document Structure	53
21.2.3	Ignoring Inputs	54
21.2.4	Structure Sharing	55
21.2.5	Global Variables	55
21.2.6	Colors	56
21.3	Limitations	56
22	Slides and Course Notes	57
22.1	Introduction	57
22.2	The User Interface	57
22.2.1	Package Options	57
22.2.2	Notes and Slides	58
22.2.3	Header and Footer Lines of the Slides	59
22.2.4	Frame Images	59
22.2.5	Colors and Highlighting	60
22.2.6	Front Matter, Titles, etc.	60
22.2.7	Excursions	60
22.2.8	Miscellaneous	60
22.3	Limitations	60
23	problem.sty: An Infrastructure for formatting Problems	61
23.1	Introduction	61
23.2	The User Interface	61
23.2.1	Package Options	61
23.2.2	Problems and Solutions	62
23.2.3	Multiple Choice Blocks	63
23.2.4	Including Problems	63
23.2.5	Reporting Metadata	63
23.3	Limitations	63
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	65
24.1	Introduction	66
24.2	The User Interface	66
24.2.1	Package and Class Options	66
24.2.2	Assignments	66
24.2.3	Typesetting Exams	66
24.2.4	Including Assignments	67
24.3	Limitations	67
IV	Implementation	69

25	STeX-Basics Implementation	70
25.1	The STeXDocument Class	70
25.2	Preliminaries	70
25.3	Messages and logging	71
25.4	Persistence	72
25.5	HTML Annotations	72
25.6	Languages	75
25.7	Activating/Deactivating Macros	76
26	STeX-MathHub Implementation	78
26.1	Generic Path Handling	78
26.2	PWD and kpsewhich	80
26.3	File Hooks and Tracking	81
26.4	MathHub Repositories	82
27	STeX-References Implementation	89
27.1	Document URIs and URLs	89
27.2	Setting Reference Targets	91
27.3	Using References	92
28	STeX-Modules Implementation	94
28.1	The module environment	97
28.2	Invoking modules	103
29	STeX-Module Inheritance Implementation	105
29.1	SMS Mode	105
29.2	Inheritance	109
30	STeX-Symbols Implementation	114
30.1	Symbol Declarations	114
30.2	Notations	120
31	STeX-Terms Implementation	129
31.1	Symbol Invocations	129
31.2	Terms	132
31.3	Notation Components	138
32	STeX-Structural Features Implementation	141
32.1	Imports with modification	141
32.2	The feature environment	145
32.3	Features	147
33	STeX-Statements Implementation	153
33.1	Definitions	153
33.2	Assertions	156
33.3	Examples	158
33.4	Logical Paragraphs	160

34 The Implementation	163
34.1 Package Options	163
34.2 Proofs	163
34.3 Justifications	169
35 \TeX-Others Implementation	171
36 \TeX-Metatheory Implementation	172
37 Tikzinput Implementation	175
38 document-structure.sty Implementation	177
38.1 The OMDoc Class	177
38.2 Class Options	177
38.3 Beefing up the <code>document</code> environment	178
38.4 Implementation: OMDoc Package	178
38.5 Package Options	178
38.6 Document Structure	180
38.7 Front and Backmatter	183
38.8 Global Variables	185
39 MiKoSlides – Implementation	186
39.1 Class and Package Options	186
39.2 Notes and Slides	188
39.3 Header and Footer Lines	192
39.4 Frame Images	193
39.5 Colors and Highlighting	194
39.6 Sectioning	195
39.7 Excursions	197
40 The Implementation	199
40.1 Package Options	199
40.2 Problems and Solutions	200
40.3 Multiple Choice Blocks	205
40.4 Including Problems	206
40.5 Reporting Metadata	207
41 Implementation: The hwexam Class	209
41.1 Class Options	209
42 Implementation: The hwexam Package	211
42.1 Package Options	211
42.2 Assignments	212
42.3 Including Assignments	215
42.4 Typesetting Exams	216
42.5 Leftovers	218

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 Primitive Symbols (The \TeX 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 1

```
\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 2

```
\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 3

```
 $\mult*{a}[\comp{\ast}]{b}$  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 4

```
\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 5

```
\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 6

```
\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 7

```
\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 8

```
\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 9

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$numseq{a,b,c}{\mathbb{R}}$
```

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

5 6

⁵EDNOTE: what about e.g. $\int \int \int f \, dx \, dy \, dz$?

⁶EDNOTE: “decompose” a-type arguments into fixed-arity operators?

Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

\TeX insert brackets thusly: Upon encountering a semantic macro (such as `\foo`), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially `\neginfprec`). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B , then A as operator precedence should be smaller than B 's argument precedences.

For example:

Example 10

```
\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 `<content>` with

`property="stex:<property>", resource="<resource>"`.

`\stex_annotate_invisible:n` adds the attributes

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

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

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

<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 `<cs>` throw an error, indicating that it is only allowed in the context of `<environments>`.

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

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

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

Chapter 10

sTEX-MathHub

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

10.1 Macros and Environments

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

10.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

10.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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



```

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

```

12.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

Test 5

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

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

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

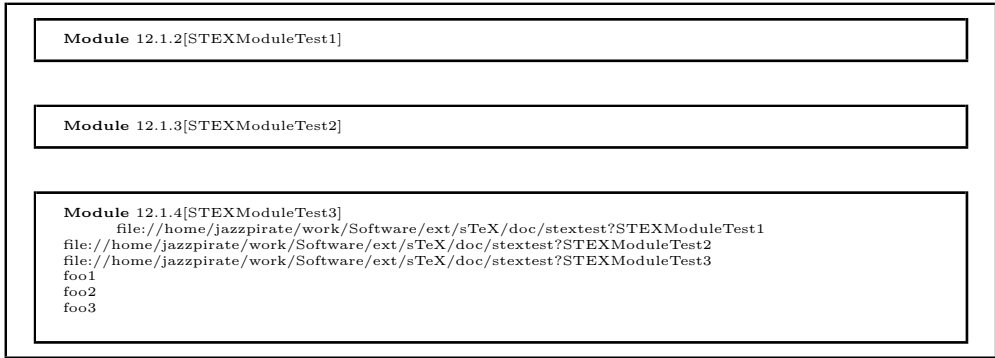
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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



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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

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

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

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

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

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

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

Test 9

```

\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:~\present\foo\\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:~\present\foo\\
Meaning:~\present\bar\\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,~} \\
All~symbols:~
\seq_use:Nn \l_stex_all_symbols_seq {,~}
\ExplSyntaxOff
\end{module}

```

Module 13.1.4[UseTest1]

Module 13.1.5[UseTest2]

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

Module 13.1.6[UseTest3]

Meaning: `>undefined<`

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

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

Test 10

```

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

```

Circular dependencies:

Module 13.1.7[CircDep1]

`>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<`

`>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<`

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

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

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

1. If `<archive-ID>` is empty:

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

`{<ns>} {<archive-ID>} {<path>} {<name>}`

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `content`-field.

Chapter 14

TEX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

<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\geq 0}`.

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 14.1.2[NotationTest]

`\symdef`

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

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

Test 13

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

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

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

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

Test 14

```

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

```

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

Test 15

```

\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle } }
$\foobar{a}{b,c,d,e,f}g$ and $\foobar{foo}{a}{b,c}g$ and $\foobar{abc}$

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[ prec=50]{ plus }{#1}{#1 \comp+ #2}
\notation[ prec=100]{ mult }{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{ac}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{ac}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle \mult{a,\plus{\frac{ab}{ac}}}$}
\end{module}

```

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

`\stex_term_custom:nn`

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

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

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

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 16

TeX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

<code>subproof</code>	The <code>pfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>method</code>	
<code>spfcases</code>	The <code>pfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>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:

`\sproofend`

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

`\sProofEndSymbol`

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

18.2.6 Configuration of the Presentation

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

EdN:8

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

The `omdoc` package is part of the $\text{\texttt{sTeX}}$ 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 \LaTeX ML transformation.

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

```

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

```

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

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

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`

`\CurrentSectionLevel`

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

21.2.3 Ignoring Inputs

`ignore`
`showignores`

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

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

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

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

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

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

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

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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. |

EdN:11

<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 \LaTeX 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 \LaTeX notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.¹²

The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
\frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\mhframeimage[fooMH/bar]{baz/foobar}
```


Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in `MathHub`.

If `baz/foobar` is the “current module”, i.e. if we are on the `MathHub` path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

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

`\mhframeimage{baz/foobar}`

22.2.5 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: 

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

In course notes, we sometimes want to point to an “excursion” – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

```
\excursion      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nomtext}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nomtext}
```

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem0.0 ()

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

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

Problem0.0 ()

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 19.)

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 19.)

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 19.)

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 19.)

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

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```



```

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

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 20.)

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 20.)

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

```

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

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

(End definition for \stex_deactivate_macro:Nn. This function is documented on page 20.)

\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 20.)

```
281 \</package>
```

Chapter 26

STEX -MathHub Implementation

```
282 <*package>
283
284 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
285
286 <@@=stex_path>
287
288 Warnings and error messages
289 \msg_new:nnn{stex}{error/norepository}{
290   No~archive~#1~found~in~#2
291 }
292 \msg_new:nnn{stex}{error/notinarchive}{
293   Not~currently~in~an~archive,~but~\detokenize{#1}~
294   needs~one!
295 }
296 \msg_new:nnn{stex}{error/nofile}{
297   \detokenize{#1}~could~not~find~file~#2
298 }
```

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
297 \cs_new_protected:Nn \stex_path_from_string:Nn {
298   \str_set:Nx \l_tmpa_str { #2 }
299   \str_if_empty:NTF \l_tmpa_str {
300     \seq_clear:N #1
301   }{
302     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
303     \sys_if_platform_windows:T{
304       \seq_clear:N \l_tmpa_tl
305       \seq_map_inline:Nn #1 {
306         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
307         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

308     }
309     \seq_set_eq:NN #1 \l_tmpa_tl
310   }
311   \stex_path_canonicalize:N #1
312 }
313 }
314 \cs_generate_variant:Nn \stex_path_from_string:Nn
315 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
316 \cs_new_protected:Nn \stex_path_to_string:NN {
317   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
318 }
319
320 \cs_new:Nn \stex_path_to_string:N {
321   \seq_use:Nn #1 /
322 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 21.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
323 \str_const:Nn \c__stex_path_dot_str {.}
324 \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.

```

325 \cs_new_protected:Nn \stex_path_canonicalize:N {
326   \seq_if_empty:NF #1 {
327     \seq_clear:N \l_tmpa_seq
328     \seq_get_left:NN #1 \l_tmpa_tl
329     \str_if_empty:NT \l_tmpa_tl {
330       \seq_put_right:Nn \l_tmpa_seq {}
331     }
332     \seq_map_inline:Nn #1 {
333       \str_set:Nn \l_tmpa_tl { ##1 }
334       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
335         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
336           \seq_if_empty:NTF \l_tmpa_seq {
337             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
338               \c__stex_path_up_str
339             }
340           }{
341             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
342             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
343               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
344                 \c__stex_path_up_str
345               }
346             }{
347               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
348             }

```

```

349     }
350   }{
351     \str_if_empty:NF \l_tmpa_tl {
352       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
353     }
354   }
355 }
356 }
357 \seq_gset_eq:NN #1 \l_tmpa_seq
358 }
359 }

```

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

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

```

360 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
361   \seq_if_empty:NTF #1 {
362     \prg_return_false:
363   }{
364     \seq_get_left:NN #1 \l_tmpa_tl
365     \str_if_empty:NTF \l_tmpa_tl {
366       \prg_return_true:
367     }{
368       \prg_return_false:
369     }
370   }
371 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

372 \str_new:N\l_stex_kpsewhich_return_str
373 \cs_new_protected:Nn \stex_kpsewhich:n {
374   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
375   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
376   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
377 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

378 \sys_if_platform_windows:TF{
379   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
380 }{
381   \stex_kpsewhich:n{-var-value~PWD}
382 }
383
384 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
385 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
386 \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 21.)

26.3 File Hooks and Tracking

387 `<@@=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

388 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

390 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

391 `\c_stex_mainfile_str`

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

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

```

392 \seq_gclear_new:N\g_stex_currentfile_seq
393 \AddToHook{file/before}{
394   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
395   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
396     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
397   }{
398     \stex_path_from_string:Nn\g_stex_currentfile_seq{
399       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
400     }
401   }
402   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
403   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
404 }
405 \AddToHook{file/after}{
406   \seq_if_empty:NF\g__stex_files_stack{
407     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
408   }
409   \seq_if_empty:NTF\g__stex_files_stack{
410     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
411   }{
412     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
413     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
414   }
415 }
```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 22.)

26.4 MathHub Repositories

```

416 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
417 \str_if_empty:NTF\mathhub{
418   \stex_kpsewhich:n{-var-value~MATHHUB}
419   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
420
421   \str_if_empty:NTF\c_stex_mathhub_str{
422     \msg_warning:nn{stex}{warning/nomathhub}
423   }{
424     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
425     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
426   }
427 }{
428   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
429   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
430     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
431       \c_stex_pwd_str/\mathhub
432     }
433   }
434   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
435   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
436 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 22.)

```

\__stex_mathhub_do_manifest:n
437 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
438   \str_set:Nx \l_tmpa_str { #1 }
439   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
440     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
441     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
442     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
443     \__stex_mathhub_find_manifest:N \l_tmpa_seq
444     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
445       \msg_error:nnxx{stex}{error/norepository}{#1}{
446         \stex_path_to_string:N \c_stex_mathhub_str
447       }
448     } {
449       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
450     }
451   }
452 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
453 \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`:

```

454 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
455   \seq_set_eq:NN \l_tmpa_seq #1
456   \bool_set_true:N \l_tmpa_bool
457   \bool_while_do:Nn \l_tmpa_bool {
458     \seq_if_empty:NTF \l_tmpa_seq {
459       \bool_set_false:N \l_tmpa_bool
460     }{
461       \file_if_exist:nTF{
462         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
463       }{
464         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
465         \bool_set_false:N \l_tmpa_bool
466       }{
467         \file_if_exist:nTF{
468           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
469         }{
470           \seq_put_right:Nn \l_tmpa_seq{META-INF}
471           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
472           \bool_set_false:N \l_tmpa_bool
473         }{
474           \file_if_exist:nTF{
475             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
476           }{
477             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
478             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
479             \bool_set_false:N \l_tmpa_bool
480           }{
481             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
482           }
483         }
484       }
485     }
486   }
487   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
488 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

489 \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:

```

490 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
491   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
492   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
493   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
494     \str_set:Nn \l_tmpa_str {##1}
495     \exp_args:NNoo \seq_set_split:Nnn
496       \l_tmpb_seq \c_colon_str \l_tmpa_str
497     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

498 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
499 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
500 }
501 \exp_args:No \str_case:nnTF \l_tmpa_tl {
502 {id} {
503 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
504 { id } \l_tmpb_tl
505 }
506 {narration-base} {
507 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
508 { narr } \l_tmpb_tl
509 }
510 {url-base} {
511 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512 { docurl } \l_tmpb_tl
513 }
514 {source-base} {
515 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516 { ns } \l_tmpb_tl
517 }
518 {ns} {
519 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520 { ns } \l_tmpb_tl
521 }
522 {dependencies} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { deps } \l_tmpb_tl
525 }
526 }{}{}
527 }{}
528 }
529 \ior_close:N \c__stex_mathhub_manifest_ior
530 }

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

531 \cs_new_protected:Nn \stex_set_current_repository:n {
532 \stex_require_repository:n { #1 }
533 \prop_set_eq:Nc \l_stex_current_repository_prop {
534 c_stex_mathhub_#1_manifest_prop
535 }
536 }

```

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

`\stex_require_repository:n`

```

537 \cs_new_protected:Nn \stex_require_repository:n {
538 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
539 \stex_debug:nn{mathhub}{Opening~archive:~#1}
540 \__stex_mathhub_do_manifest:n { #1 }
541 \exp_args:Nx \stex_add_to_sms:n {
542 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
543 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
544 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

545     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
546     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
547   }
548 }
549 }
550 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

551 %\prop_new:N \l_stex_current_repository_prop
552
553 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
554 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
555   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
556 } {
557   \__stex_mathhub_parse_manifest:n { main }
558   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
559   \l_tmpa_str
560   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
561   \c_stex_mathhub_main_manifest_prop
562   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
563   \stex_debug:nn{mathhub}{Current~repository:~
564   \prop_item:Nn \l_stex_current_repository_prop {id}
565   }
566 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 22.)

`\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.

```

567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \prop_if_exist:NTF \l_stex_current_repository_prop {
572       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
573       \exp_args:Ne \l_tmpa_cs{
574         \prop_item:Nn \l_stex_current_repository_prop { id }
575       }
576     }{
577       \l_tmpa_cs{}
578     }
579   }{
580     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
581     \stex_require_repository:n \l_tmpa_str
582     \str_set:Nx \l_tmpa_str { #1 }
583     \exp_args:Nne \use:nn {
584       \stex_set_current_repository:n \l_tmpa_str
585       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
586     }{
587       \stex_debug:nn{mathhub}{switching~back~to:~
588       \prop_if_exist:NTF \l_stex_current_repository_prop {
589         \prop_item:Nn \l_stex_current_repository_prop { id }::~

```

```

590         \meaning\l_stex_current_repository_prop
591     }{
592         no~repository
593     }
594 }
595 \prop_if_exist:NTF \l_stex_current_repository_prop {
596     \stex_set_current_repository:n {
597         \prop_item:Nn \l_stex_current_repository_prop { id }
598     }
599 }{
600     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
601 }
602 }
603 }
604 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn
605 \newif \ifinputref \inputreffalse
606
607 \cs_new_protected:Nn \stex_mhinput:nn {
608     \stex_in_repository:nn {#1} {
609         \ifinputref
610             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
611         \else
612             \inputreftrue
613             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
614             \inputreffalse
615         \fi
616     }
617 }
618 \NewDocumentCommand \mhinput { 0{} m}{
619     \stex_mhinput:nn{ #1 }{ #2 }
620 }
621
622 \cs_new_protected:Nn \stex_inputref:nn {
623     \stex_in_repository:nn {#1} {
624         \bool_lazy_any:nTF {
625             {\rustex_if_p:} {\latexml_if_p:}
626         } {
627             \str_clear:N \l_tmpa_str
628             \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
629                 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
630             }
631             \stex_annotate_invisible:nnn{inputref}{
632                 \l_tmpa_str / #2
633             }{}
634         }{
635             \begingroup
636             \inputreftrue
637             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
638             \endgroup
639         }

```

```

640 }
641 }
642
643 \NewDocumentCommand \inputref { 0{ } m }{
644   \stex_inputref:nn{ #1 }{ #2 }
645 }
646
647 \cs_new_protected:Nn \stex_mhbibresource:nn {
648   \stex_in_repository:nn {#1} {
649     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
650   }
651 }
652 \newcommand\addmhbibresource[2][]{
653   \stex_mhbibresource:nn{ #1 }{ #2 }
654 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page 23.)

`\mhpath`

```

655 \def \mhpath #1 #2 {
656   \exp_args:Ne \str_if_eq:nnTF{#1}{-}{
657     \c_stex_mathhub_str /
658     \prop_item:Nn \l_stex_current_repository_prop { id }
659     / source / #2
660   }{
661     \c_stex_mathhub_str / #1 / source / #2
662   }
663 }

```

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

`\libinput`

```

664 \cs_new_protected:Npn \libinput #1 {
665   \prop_if_exist:NF \l_stex_current_repository_prop {
666     \msg_error:nnn{stex}{error/notinarchive}\libinput
667   }
668   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
669     \msg_error:nnn{stex}{error/notinarchive}\libinput
670   }
671   \bool_set_false:N \l_tmpa_bool
672   \tl_clear:N \l_tmpa_tl
673   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
674   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
675   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
676   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
677     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
678     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
679       / meta-inf / lib / #1.tex}{
680       \bool_set_true:N \l_tmpa_bool
681       \tl_put_right:Nx \l_tmpa_tl {
682         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
683           / meta-inf / lib / #1.tex}
684       }
685     }{}

```

```

686 }
687 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
688 / \l_tmpa_str / lib / #1.tex
689 }{
690   \bool_set_true:N \l_tmpa_bool
691   \tl_put_right:Nx \l_tmpa_tl {
692     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
693       / \l_tmpa_str / lib / #1.tex}
694   }
695 }{}
696 \bool_if:NF \l_tmpa_bool {
697   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
698 }
699 \l_tmpa_tl
700 }

```

(End definition for \libinput. This function is documented on page 23.)

```

701 </package>

```

Chapter 27

STEX -References Implementation

```
702 <*package>
703
704 %%%%%%%%%% references.dtx %%%%%%%%%%
705
706 %\RequirePackage{hyperref}
707 %\RequirePackage{cleveref}
708 <@@=stex_refs>
709
710 Warnings and error messages
711
712 \iow_new:N \c__stex_refs_refs_iow
713 \AddToHook{begindocument}{
714   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
715 }
716 \AddToHook{enddocument}{
717   \iow_close:N \c__stex_refs_refs_iow
718 }
719
720 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
721
722 \NewDocumentCommand \STEXreftitle { m } {
723   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
724 }
```

27.1 Document URIs and URLs

```
723 \seq_new:N \g__stex_refs_all_refs_seq
724
725 \str_new:N \l_stex_current_docns_str
726
727 \cs_new_protected:Nn \stex_get_document_uri: {
728   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
729   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
730   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
731   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

732 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
733
734 \str_clear:N \l_tmpa_str
735 \prop_if_exist:NT \l_stex_current_repository_prop {
736   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
737     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
738   }
739 }
740
741 \str_if_empty:NTF \l_tmpa_str {
742   \str_set:Nx \l_stex_current_docns_str {
743     file:/\stex_path_to_string:N \l_tmpa_seq
744   }
745 }{
746   \bool_set_true:N \l_tmpa_bool
747   \bool_while_do:Nn \l_tmpa_bool {
748     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
749     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
750       {source} { \bool_set_false:N \l_tmpa_bool }
751     }{}{
752       \seq_if_empty:NT \l_tmpa_seq {
753         \bool_set_false:N \l_tmpa_bool
754       }
755     }
756   }
757
758   \seq_if_empty:NTF \l_tmpa_seq {
759     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
760   }{
761     \str_set:Nx \l_stex_current_docns_str {
762       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
763     }
764   }
765 }
766 }
767
768 \str_new:N \l_stex_current_docurl_str
769 \cs_new_protected:Nn \stex_get_document_url: {
770   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
771   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
772   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
773   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
774   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
775
776   \str_clear:N \l_tmpa_str
777   \prop_if_exist:NT \l_stex_current_repository_prop {
778     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
779       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
780         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
781       }
782     }
783
784     \str_if_empty:NTF \l_tmpa_str {
785       \str_set:Nx \l_stex_current_docurl_str {

```



```

786     file:/\stex_path_to_string:N \l_tmpa_seq
787   }
788 }{
789   \bool_set_true:N \l_tmpa_bool
790   \bool_while_do:Nn \l_tmpa_bool {
791     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
792     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
793       {source} { \bool_set_false:N \l_tmpa_bool }
794     }{}{
795       \seq_if_empty:NT \l_tmpa_seq {
796         \bool_set_false:N \l_tmpa_bool
797       }
798     }
799   }
800
801   \seq_if_empty:NTF \l_tmpa_seq {
802     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
803   }{
804     \str_set:Nx \l_stex_current_docurl_str {
805       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
806     }
807   }
808 }
809 }

```

27.2 Setting Reference Targets

```

810 \str_const:Nn \c__stex_refs_url_str{URL}
811 \str_const:Nn \c__stex_refs_ref_str{REF}
812 % @currentlabel -> number
813 % @currentlabelname -> title
814 % @currentHref -> name.number <- id of some kind
815 % \theH# -> \arabic{section}
816 % \the# -> number
817 % \hyper@makecurrent{#}
818 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
819   \stex_get_document_uri:
820   \str_set:Nx \l_tmpa_str { #1 }
821   \str_if_empty:NT \l_tmpa_str {
822     \int_zero:N \l_tmpa_int
823     \bool_set_true:N \l_tmpa_bool
824     \bool_while_do:Nn \l_tmpa_bool {
825       \cs_if_exist:cTF {
826         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
827       }{
828         \int_incr:N \l_tmpa_int
829       }{
830         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
831         \bool_set_false:N \l_tmpa_bool
832       }
833     }
834   }
835   \str_set:Nx \l_tmpa_str {
836     \l_stex_current_docns_str??\l_tmpa_str

```

```

837 }
838 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
839 \stex_if_smsmode:TF {
840   \stex_get_document_url:
841   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
842   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
843 }{
844   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
845   \exp_args:Nx\label{sref_\l_tmpa_str}
846
847   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
848   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
849 }
850 }
851 \cs_new_protected:Npn \stexauxadddocref #1 {
852   \str_set:Nx \l_tmpa_str {#1}
853   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
854   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
855 }
856 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
857   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
858 }

```

27.3 Using References

```

859 \str_new:N \l__stex_refs_indocument_str
860 \keys_define:nn { stex / sref } {
861   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
862   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
863   pre           .tl_set:N = \l__stex_refs_pre_tl ,
864   post          .tl_set:N = \l__stex_refs_post_tl ,
865   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
866 }
867
868 \bool_new:N \c__stex_refs_hyperref_bool
869 \bool_set_false:N \c__stex_refs_hyperref_bool
870 \AddToHook{begindocument}{
871   \ifpackageloaded{hyperref}{
872     \bool_set_true:N \c__stex_refs_hyperref_bool
873   }{}
874 }
875
876
877 \cs_new_protected:Nn \__stex_refs_args:n {
878   \tl_clear:N \l__stex_refs_linktext_tl
879   \tl_clear:N \l__stex_refs_fallback_tl
880   \tl_clear:N \l__stex_refs_pre_tl
881   \tl_clear:N \l__stex_refs_post_tl
882   \str_clear:N \l__stex_refs_repo_str
883   \keys_set:nn { stex / sref } { #1 }
884 }
885
886 \NewDocumentCommand \sref { 0{} m}{
887   \__stex_refs_args:n { #1 }

```

```

888 \str_if_empty:NTF \l__stex_refs_indocument_str {
889   \str_set:Nn \l_tmpa_str { #2 }
890   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
891   \tl_set:Nn \l_tmpa_tl {
892     \l__stex_refs_fallback_tl
893   }
894   \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
895     \str_set:Nn \l_tmpb_str { ##1 }
896     \str_if_eq:eeT { \l_tmpa_str } {
897       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
898     } {
899       \seq_map_break:n {
900         \tl_set:Nn \l_tmpa_tl {
901           % doc uri in \l_tmpb_str
902           \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
903           \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
904             % reference
905             \cs_if_exist:cTF{autoref}{
906               \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
907             }{
908               \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
909             }
910           }{
911             % URL
912             \if_bool:N \c__stex_refs_hyperref_bool {
913               \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
914             }{
915               \l__stex_refs_fallback_tl
916             }
917           }
918         }
919       }
920     }
921   }
922   \l_tmpa_tl
923 }{
924   % TODO
925 }
926 }
927
928 </package>

```

Chapter 28

STEX -Modules Implementation

```
929 <*package>
930
931 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
932
933 <@@=stex_modules>
934
935   Warnings and error messages
936   \msg_new:nnn{stex}{error/unknownmodule}{
937     No~module~#1~found
938   }
939   \msg_new:nnn{stex}{error/syntax}{
940     Syntax~error:~#1
941   }
942   \msg_new:nnn{stex}{error/siglanguage}{
943     Module~#1~declares~signature~#2,~but~does~not~
944     declare~its~language
945   }
946   \msg_new:nnn{stex}{error/conclictingmodules}{
947     Comflicting~imports~for~module~#1
948   }
949
950 \l_stex_current_module_str The current module:
951
952   \str_new:N \l_stex_current_module_str
953
954 (End definition for \l_stex_current_module_str. This variable is documented on page 25.)
955
956 \l_stex_all_modules_seq Stores all available modules
957
958   \seq_new:N \l_stex_all_modules_seq
959
960 (End definition for \l_stex_all_modules_seq. This variable is documented on page 25.)
961
962 \stex_if_in_module_p:
963 \stex_if_in_module:TF
964
965   \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
966     \str_if_empty:NTF \l_stex_current_module_str
967       \prg_return_false: \prg_return_true:
968   }
```

(End definition for `\stex_if_in_module:TF`. This function is documented on page 26.)

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

```
954 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
955   \prop_if_exist:cTF { c_stex_module_#1_prop }
956   \prg_return_true: \prg_return_false:
957 }
```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 26.)

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
958 \cs_new_protected:Nn \stex_add_to_current_module:n {
959   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
960 }
961 \cs_new_protected:Npn \STEXexport {
962   \begingroup
963   \newlinechar=-1\relax
964   \endlinechar=-1\relax
965   %\catcode'\ = 9\relax
966   \expandafter\endgroup\STEXexport:n
967 }
968 \cs_new_protected:Nn \STEXexport:n {
969   \ignorespaces #1
970   \stex_add_to_current_module:n { \ignorespaces #1 }
971   \stex_smsmode_set_codes:
972 }
973 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 26.)

`\stex_add_constant_to_current_module:n`

```
974 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
975   \str_set:Nx \l_tmpa_str { #1 }
976   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
977 }
978
979 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
980 % \str_set:Nx \l_tmpa_str { #1 }
981 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_fields} { \l_tmpa_str }
982 %}
```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 26.)

`\stex_collect_imports:n`

```
983 \cs_new_protected:Nn \stex_collect_imports:n {
984   \seq_clear:N \l_stex_collect_imports_seq
985   \__stex_modules_collect_imports:n {#1}
986 }
987 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
988   \seq_map_inline:cn {c_stex_module_#1_imports} {
989     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
990       \__stex_modules_collect_imports:n { ##1 }
991     }
992 }
```

```

992 }
993 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
994   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
995 }
996 }

```

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

`\stex_add_import_to_current_module:n`

```

997 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
998   \str_set:Nx \l_tmpa_str { #1 }
999   \exp_args:Nno
1000   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1001     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1002   }
1003 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 26.)

`\stex_modules_compute_namespace:nN`

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

```

1004 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1005   \str_set:Nx \l_tmpa_str { #1 }
1006   \seq_set_eq:NN \l_tmpa_seq #2
1007   % split off file extension
1008   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1009   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1010   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1011   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1012
1013   \bool_set_true:N \l_tmpa_bool
1014   \bool_while_do:Nn \l_tmpa_bool {
1015     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1016     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1017       {source} { \bool_set_false:N \l_tmpa_bool }
1018     }{}{
1019       \seq_if_empty:NT \l_tmpa_seq {
1020         \bool_set_false:N \l_tmpa_bool
1021       }
1022     }
1023   }
1024
1025   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1026   \str_if_empty:NTF \l_stex_modules_subpath_str {
1027     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1028   }{
1029     \str_set:Nx \l_stex_modules_ns_str {
1030       \l_tmpa_str/\l_stex_modules_subpath_str
1031     }
1032   }
1033 }

```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 26.)

Stores its return values in:

```
\l_stex_modules_ns_str
\l_stex_modules_subpath_str
```

```
1034 \str_new:N \l_stex_modules_ns_str
1035 \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.

```
1036 \cs_new_protected:Nn \stex_modules_current_namespace: {
1037   \str_clear:N \l_stex_modules_subpath_str
1038   \prop_if_exist:NTF \l_stex_current_repository_prop {
1039     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1040     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1041   }{
1042     % split off file extension
1043     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1044     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1045     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1046     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1047     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1048     \str_set:Nx \l_stex_modules_ns_str {
1049       file:/\stex_path_to_string:N \l_tmpa_seq
1050     }
1051   }
1052 }
```

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

28.1 The module environment

module arguments:

```
1053 \keys_define:nn { stex / module } {
1054   title      .str_set_x:N = \l_stex_module_title_str ,
1055   ns         .str_set_x:N = \l_stex_module_ns_str ,
1056   lang       .str_set_x:N = \l_stex_module_lang_str ,
1057   sig        .str_set_x:N = \l_stex_module_sig_str ,
1058   creators   .str_set_x:N = \l_stex_module_creators_str ,
1059   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1060   meta       .str_set_x:N = \l_stex_module_meta_str ,
1061   srccite    .str_set_x:N = \l_stex_module_srccite_str
1062 }
1063
1064 \cs_new_protected:Nn \__stex_modules_args:n {
1065   \str_clear:N \l_stex_module_title_str
1066   \str_clear:N \l_stex_module_ns_str
1067   \str_clear:N \l_stex_module_lang_str
1068   \str_clear:N \l_stex_module_sig_str
1069   \str_clear:N \l_stex_module_creators_str
1070   \str_clear:N \l_stex_module_contributors_str
1071   \str_clear:N \l_stex_module_meta_str
1072   \str_clear:N \l_stex_module_srccite_str
1073   \keys_set:nn { stex / module } { #1 }
```

```

1074 }
1075
1076 % module parameters here? In the body?
1077

```

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

```

1078 \cs_new_protected:Nn \stex_module_setup:nn {
1079   \str_set:Nx \l_stex_module_name_str { #2 }
1080   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1081 \stex_if_in_module:TF {
1082   % Nested module
1083   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1084   { ns } \l_stex_module_ns_str
1085   \str_set:Nx \l_stex_module_name_str {
1086     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1087     { name } / \l_stex_module_name_str
1088   }
1089 }{
1090   % not nested:
1091   \str_if_empty:NT \l_stex_module_ns_str {
1092     \stex_modules_current_namespace:
1093     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1094     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1095     / { \l_stex_module_ns_str }
1096     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1097     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1098       \str_set:Nx \l_stex_module_ns_str {
1099         \stex_path_to_string:N \l_tmpa_seq
1100       }
1101     }
1102   }
1103 }

```

Next, we determine the language of the module:

```

1104 \str_if_empty:NT \l_stex_module_lang_str {
1105   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1106   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1107   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1108   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1109   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1110     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1111       inferred~from~file~name}
1112     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1113   }
1114 }
1115
1116 \str_if_empty:NF \l_stex_module_lang_str {
1117   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1118   \l_tmpa_str {
1119     \ltx@ifpackageloaded{babel}{
1120       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```



```

1121     }{}
1122   } {
1123     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1124   }
1125 }

```

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

```

1126 \str_if_empty:NTF \l_stex_module_sig_str {
1127   \exp_args:Nnx \prop_gset_from_keyval:cn {
1128     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1129   } {
1130     name      = \l_stex_module_name_str ,
1131     ns        = \l_stex_module_ns_str ,
1132     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1133     lang      = \l_stex_module_lang_str ,
1134     sig       = \l_stex_module_sig_str ,
1135     meta      = \l_stex_module_meta_str
1136   }
1137   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1138   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1139   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1140   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1141   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1142 \str_if_empty:NT \l_stex_module_meta_str {
1143   \str_set:Nx \l_stex_module_meta_str {
1144     \c_stex_metatheory_ns_str ? Metatheory
1145   }
1146 }
1147 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1148   \bool_set_true:N \l_stex_in_meta_bool
1149   \exp_args:Nx \stex_add_to_current_module:n {
1150     \bool_set_true:N \l_stex_in_meta_bool
1151     \stex_activate_module:n {\l_stex_module_meta_str}
1152     \bool_set_false:N \l_stex_in_meta_bool
1153   }
1154   \stex_activate_module:n {\l_stex_module_meta_str}
1155   \bool_set_false:N \l_stex_in_meta_bool
1156 }
1157 }{
1158   \str_if_empty:NT \l_stex_module_lang_str {
1159     \msg_error:nnxx{stex}{error/siglanguage}{
1160       \l_stex_module_ns_str?\l_stex_module_name_str
1161     }{\l_stex_module_sig_str}
1162   }
1163
1164   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1165   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1166   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1167   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1168   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1169   \str_set:Nx \l_tmpa_str {

```

```

1170     \stex_path_to_string:N \l_tmpa_seq /
1171     \l_tmpa_str . \l_stex_module_sig_str .tex
1172 }
1173 \IfFileExists \l_tmpa_str {
1174     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1175         \seq_clear:N \l_stex_all_modules_seq
1176         %\prop_clear:N \l_stex_current_module_prop
1177         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1178         \input { \l_tmpa_str }
1179     }
1180 }{
1181     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1182 }
1183 \stex_activate_module:n {
1184     \l_stex_module_ns_str ? \l_stex_module_name_str
1185 }
1186 %\prop_set_eq:Nc \l_stex_current_module_prop {
1187 % c_stex_module_
1188 % \l_stex_module_ns_str ?
1189 % \l_stex_module_name_str
1190 % _prop
1191 %}
1192 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1193 }
1194 }

```

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

module The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1195 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1196     \stex_reactivate_macro:N \STEXexport
1197     \stex_reactivate_macro:N \importmodule
1198     \stex_reactivate_macro:N \symdecl
1199     \stex_reactivate_macro:N \notation
1200     \stex_reactivate_macro:N \symdef
1201     \stex_module_setup:nn{#1}{#2}
1202 }
1203 \stex_debug:nn{modules}{
1204     New~module:\\
1205     Namespace:~\l_stex_module_ns_str\\
1206     Name:~\l_stex_module_name_str\\
1207     Language:~\l_stex_module_lang_str\\
1208     Signature:~\l_stex_module_sig_str\\
1209     Metatheory:~\l_stex_module_meta_str\\
1210     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1211 }
1212 }
1213 \seq_put_right:Nx \l_stex_all_modules_seq {
1214     \l_stex_module_ns_str ? \l_stex_module_name_str
1215 }
1216 }
1217 % \seq_gput_right:Nx \g_stex_modules_in_file_seq

```

```

1218 %      { \l_stex_module_ns_str ? \l_stex_module_name_str }
1219
1220 \stex_if_smsmode:TF {
1221   \stex_smsmode_set_codes:
1222 } {
1223   \begin{stex_annotate_env} {theory} {
1224     \l_stex_module_ns_str ? \l_stex_module_name_str
1225   }
1226
1227   \stex_annotate_invisible:nnn{header}{} {
1228     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1229     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1230     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1231       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1232     }
1233   }
1234 }
1235 % TODO: Inherit metatheory for nested modules?
1236 }
1237 \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}`

```

1238 \cs_new_protected:Nn \_stex_modules_end_module: {
1239 %   \str_set:Nx \l_tmpa_str {
1240 %     c_stex_module_
1241 %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1242 %     \prop_item:Nn \l_stex_current_module_prop { name }
1243 %     _prop
1244 %   }
1245 %^^A \prop_new:c { \l_tmpa_str }
1246 %   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1247   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1248 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1249 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1250 \NewDocumentEnvironment { @module } { 0 } { m } {
1251   \par
1252   \_stex_modules_begin_module:nn{#1}{#2}
1253 } {
1254   \_stex_modules_end_module:
1255   \stex_if_smsmode:TF {
1256 %     \exp_args:Nx \stex_add_to_sms:n {
1257 %       \prop_gset_from_keyval:cn {
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 %         name          = \prop_item:cn { \l_tmpa_str } { name } ,

```

```

1264 %      ns      = \prop_item:cn { \l_tmpa_str } { ns } ,
1265 %      file     = \prop_item:cn { \l_tmpa_str } { file } ,
1266 %      lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
1267 %      sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
1268 %      meta     = \prop_item:cn { \l_tmpa_str } { meta }
1269 %    }
1270 %  }
1271 }{
1272   \end{stex_annotate_env}
1273 }
1274 }

```

\stex_modules_heading: Code for document headers

```

1275 \cs_if_exist:NTF \thesection {
1276   \newcounter{module}[section]
1277 }{
1278   \newcounter{module}
1279 }
1280
1281 \bool_if:NT \c_stex_showmods_bool {
1282   \latexml_if:F { \RequirePackage{mdframed} }
1283 }
1284
1285 \cs_new_protected:Nn \stex_modules_heading: {
1286   \stepcounter{module}
1287   \par
1288   \bool_if:NT \c_stex_showmods_bool {
1289     \noindent{\textbf{Module} ~
1290       \cs_if_exist:NT \thesection {\thesection.}
1291       \themodule ~ [\l_stex_module_name_str]
1292     }
1293     \str_if_empty:NTF \l_stex_module_title_str {
1294       }{
1295         \quad(\l_stex_module_title_str)\hfill
1296       }\par
1297     }
1298     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1299     % TODO
1300     \stex_ref_new_doc_target:n \l_stex_module_name_str
1301   }

```

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

Finally:

```

1302 \NewDocumentEnvironment { module } { 0{} m } {
1303   \bool_if:NT \c_stex_showmods_bool {
1304     \begin{mdframed}
1305   }
1306   \begin{@module}[#1]{#2}
1307   \stex_modules_heading:
1308 }{
1309   \end{@module}
1310   \bool_if:NT \c_stex_showmods_bool {
1311     \end{mdframed}
1312   }

```

1313 }

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1314 \NewDocumentCommand \STEXModule { m } {
1315   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1316   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1317   \tl_set:Nn \l_tmpa_tl {
1318     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1319   }
1320   \seq_map_inline:Nn \l_stex_all_modules_seq {
1321     \str_set:Nn \l_tmpb_str { ##1 }
1322     \str_if_eq:eeT { \l_tmpa_str } {
1323       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1324     } {
1325       \seq_map_break:n {
1326         \tl_set:Nn \l_tmpa_tl {
1327           \stex_invoke_module:n { ##1 }
1328         }
1329       }
1330     }
1331   }
1332   \l_tmpa_tl
1333 }
1334
1335 \cs_new_protected:Nn \stex_invoke_module:n {
1336   \stex_debug:nn{modules}{Invoking~module~#1}
1337   \peek_charcode_remove:NTF ! {
1338     \__stex_modules_invoke_uri:nN { #1 }
1339   } {
1340     \peek_charcode_remove:NTF ? {
1341       \__stex_modules_invoke_symbol:nn { #1 }
1342     } {
1343       \msg_error:nnx{stex}{error/syntax}{
1344         ?~or~!~expected~after~
1345         \c_backslash_str STEXModule{#1}
1346       }
1347     }
1348   }
1349 }
1350
1351 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1352   \str_set:Nn #2 { #1 }
1353 }
1354
1355 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1356   \stex_invoke_symbol:n{#1?#2}
1357 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 28.)

`\stex_activate_module:n`

```
1358 \bool_new:N \l_stex_in_meta_bool
1359 \bool_set_false:N \l_stex_in_meta_bool
1360 \cs_new_protected:Nn \stex_activate_module:n {
1361   \stex_debug:nn{modules}{Activating~module~#1}
1362   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1363     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1364   }
1365   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1366     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1367     \use:c{ c_stex_module_#1_code }
1368   }
1369 }
```

(End definition for \stex_activate_module:n. This function is documented on page 29.)

```
1370 </package>
```

Chapter 29

sTeX -Module Inheritance Implementation

```
1371 <*package>
1372
1373 %%%%%%%%% inheritance.dtx %%%%%%%%%
1374
```

29.1 SMS Mode

```
1375 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1376 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1377 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1378 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1379
1380 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1381   \makeatletter
1382   \makeatother
1383   \ExplSyntaxOn
1384   \ExplSyntaxOff
1385 }
1386
1387 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1388   \symdef
1389   \importmodule
1390   \notation
1391   \symdecl
1392   \STEXexport
1393 }
1394
1395 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1396   \tl_to_str:n {
1397     module,
1398     @module
```

```
1399 }
1400 }
```

(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 30.)

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

```
1401 \bool_new:N \g__stex_smsmode_bool
1402 \bool_set_false:N \g__stex_smsmode_bool
1403 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1404   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1405 }
```

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

```
\_stex_smsmode_if_catcodes_p:
```

Checks whether the SMS mode category code scheme is active.

```
\_stex_smsmode_if_catcodes:TF
```

```
1406 \bool_new:N \g__stex_smsmode_catcode_bool
1407 \bool_set_false:N \g__stex_smsmode_catcode_bool
1408 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1409   \bool_if:NTF \g__stex_smsmode_catcode_bool
1410   \prg_return_true: \prg_return_false:
1411 }
```

(End definition for `_stex_smsmode_if_catcodes:TF`.)

```
\stex_smsmode_set_codes:
```

```
1412 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1413   \stex_if_smsmode:T {
1414     \_stex_smsmode_if_catcodes:F {
1415       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1416       \exp_after:wN \char_gset_active_eq:NN
1417       \c_backslash_str \_stex_smsmode_cs:
1418       \tex_global:D \char_set_catcode_active:N \
1419       \tex_global:D \char_set_catcode_other:N $
1420       \tex_global:D \char_set_catcode_other:N ^
1421       \tex_global:D \char_set_catcode_other:N _
1422       \tex_global:D \char_set_catcode_other:N &
1423       \tex_global:D \char_set_catcode_other:N ##
1424     }
1425   }
1426 } \iffalse $ \fi % to make syntax highlighting work again
```

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

```
\_stex_smsmode_unset_codes:
```

Sets category code scheme back from the one used in SMS mode.

```
1427 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1428   \_stex_smsmode_if_catcodes:T {
1429     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1430     \exp_after:wN \tex_global:D \exp_after:wN
1431     \char_set_catcode_escape:N \c_backslash_str
1432     \tex_global:D \char_set_catcode_math_toggle:N $
1433     \tex_global:D \char_set_catcode_math_superscript:N ^
1434     \tex_global:D \char_set_catcode_math_subscript:N _
1435     \tex_global:D \char_set_catcode_alignment:N &
1436     \tex_global:D \char_set_catcode_parameter:N ##
1437   }
1438 } \iffalse $ \fi % to make syntax highlighting work again
```


(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1439 \cs_new_protected:Nn \stex_in_smsmode:nn {
1440   \vbox_set:Nn \l_tmpa_box {
1441     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1442     \bool_gset_true:N \g__stex_smsmode_bool
1443     \stex_smsmode_set_codes:
1444     #2
1445     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1446     \stex_if_smsmode:F {
1447       \__stex_smsmode_unset_codes:
1448     }
1449   }
1450   \box_clear:N \l_tmpa_box
1451 }

```

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

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

```

1452 \cs_new_protected:Nn \_stex_smsmode_cs: {
1453   \str_clear:N \l_tmpa_str
1454   \peek_analysis_map_inline:n {
1455     % #1: token (one expansion)
1456     % #2: charcode
1457     % #3 catcode
1458     \token_if_eq_charcode:NNTF ##3 B {
1459       % token is a letter
1460       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1461     } {
1462       \str_if_empty:NTF \l_tmpa_str {
1463         % we don't allow (or need) single non-letter CSs
1464         % for now
1465         \peek_analysis_map_break:
1466       }{
1467         \str_if_eq:onTF \l_tmpa_str { begin } {
1468           \peek_analysis_map_break:n {
1469             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1470           }
1471         } {
1472           \str_if_eq:onTF \l_tmpa_str { end } {
1473             \peek_analysis_map_break:n {
1474               \exp_after:wN \_stex_smsmode_checkend:n ##1
1475             }
1476           } {
1477             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1478             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1479             \g_stex_smsmode_allowedmacros_tl
1480             { \use:c{\l_tmpa_str} } {
1481               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1482               \peek_analysis_map_break:n {
1483                 \exp_after:wN \l_tmpa_tl ##1
1484               }

```

```

1485     } {
1486         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1487         \g_stex_smsmode_allowedmacros_escape_tl
1488         { \use:c{\l_tmpa_str} } {
1489             \__stex_smsmode_unset_codes:
1490             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1491             % TODO \__stex_smsmode_rescan_cs:
1492             % \int_compare:nNnTF {##2} = {92} {
1493             %     \peek_analysis_map_break:n {
1494             %         \__stex_smsmode_unset_codes:
1495             %         \__stex_smsmode_rescan_cs:
1496             %     }
1497             % } {
1498             %     \peek_analysis_map_break:n {
1499             %         \exp_after:wN \l_tmpa_tl ##1
1500             %     }
1501             % }
1502         } {
1503             \int_compare:nNnTF {##2} = {92} {
1504                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1505             }{
1506                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1507             }
1508         }
1509     }
1510 }
1511 }
1512 }
1513 }
1514 }
1515 }

```

(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:

```

1516 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1517     \str_clear:N \l_tmpb_str
1518     \peek_analysis_map_inline:n {
1519         \token_if_eq_charcode:NNTF ##3 B {
1520             % token is a letter
1521             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1522         } {
1523             \peek_analysis_map_break:n {
1524                 \exp_after:wN \use:c \exp_after:wN {
1525                     \exp_after:wN \l_tmpa_str\exp_after:wN
1526                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1527             }
1528         }
1529     }
1530 }

```

(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.

```

1531 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1532   \str_set:Nn \l_tmpa_str { #1 }
1533   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1534     \__stex_smsmode_unset_codes:
1535     \begin{#1}
1536   }
1537 }
```

(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.

```

1538 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1539   \str_set:Nn \l_tmpa_str { #1 }
1540   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1541     \end{#1}
1542   }
1543 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1544 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1545 \cs_new_protected:Nn \stex_import_module_uri:nn {
1546   \str_set:Nx \l_stex_import_archive_str { #1 }
1547   \str_set:Nn \l_stex_import_path_str { #2 }
1548
1549   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1550   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1551   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1552
1553   \stex_modules_current_namespace:
1554   \bool_lazy_all:nTF {
1555     {\str_if_empty_p:N \l_stex_import_archive_str}
1556     {\str_if_empty_p:N \l_stex_import_path_str}
1557     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1558   }{
1559     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1560     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1561   }{
1562     \str_if_empty:NT \l_stex_import_archive_str {
1563       \prop_if_exist:NT \l_stex_current_repository_prop {
1564         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1565       }
1566     }
1567     \str_if_empty:NTF \l_stex_import_archive_str {
1568       \str_if_empty:NF \l_stex_import_path_str {
1569         \str_set:Nx \l_stex_import_ns_str {
1570           \l_stex_module_ns_str / \l_stex_import_path_str
1571         }
1572       }
1573     }
```

```

1573   }{
1574     \stex_require_repository:n \l_stex_import_archive_str
1575     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1576     \l_stex_import_ns_str
1577     \str_if_empty:NF \l_stex_import_path_str {
1578       \str_set:Nx \l_stex_import_ns_str {
1579         \l_stex_import_ns_str / \l_stex_import_path_str
1580       }
1581     }
1582   }
1583 }
1584 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 33.)

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1585 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1586 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1587 \str_new:N \l_stex_import_path_str
                           1588 \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>}
1589 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1590   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1591
1592     % archive
1593     \str_set:Nx \l_tmpa_str { #2 }
1594     \str_if_empty:NTF \l_tmpa_str {
1595       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1596     } {
1597       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1598       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1599       \seq_put_right:Nn \l_tmpa_seq { source }
1600     }
1601
1602     % path
1603     \str_set:Nx \l_tmpb_str { #3 }
1604     \str_if_empty:NTF \l_tmpb_str {
1605       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1606
1607       \ltx@ifpackageloaded{babel} {
1608         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1609           { \language } \l_tmpb_str {
1610           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1611         }
1612       } {
1613         \str_clear:N \l_tmpb_str
1614       }
1615
1616       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1617       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1618         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1619 }{
1620   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1621   \IfFileExists{ \l_tmpa_str.tex }{
1622     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1623   }{
1624     % try english as default
1625     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1626     \IfFileExists{ \l_tmpa_str.en.tex }{
1627       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1628     }{
1629       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1630     }
1631   }
1632 }
1633
1634 } {
1635   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1636   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1637
1638   \ltx@ifpackageloaded{babel} {
1639     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1640       { \language } \l_tmpb_str {
1641       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1642     }
1643   } {
1644     \str_clear:N \l_tmpb_str
1645   }
1646
1647   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1648
1649   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1650   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1651     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1652   }{
1653     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1654     \IfFileExists{ \l_tmpa_str/#4.tex }{
1655       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1656     }{
1657       % try english as default
1658       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1659       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1660         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1661       }{
1662         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1663         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1664           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1665         }{
1666           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1667           \IfFileExists{ \l_tmpa_str.tex }{
1668             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1669           }{
1670             % try english as default
1671             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1672             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1673         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1674     }{
1675         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1676     }
1677 }
1678 }
1679 }
1680 }
1681 }
1682 }
1683
1684 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1685     \seq_clear:N \l_stex_all_modules_seq
1686     \str_clear:N \l_stex_current_module_str
1687     \str_set:Nx \l_tmpb_str { #2 }
1688     \str_if_empty:NF \l_tmpb_str {
1689         \stex_set_current_repository:n { #2 }
1690     }
1691     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1692     \input { \g__stex_importmodule_file_str }
1693 }
1694
1695 \stex_if_module_exists:nF { #1 ? #4 } {
1696     \msg_error:nnx{stex}{error/unknownmodule}{
1697         #1?#4~(in~file~\g__stex_importmodule_file_str)
1698     }
1699 }
1700 }
1701 \stex_activate_module:n { #1 ? #4 }
1702 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 33.)

`\importmodule`

```

1703 \NewDocumentCommand \importmodule { 0{} m } {
1704     \stex_import_module_uri:nn { #1 } { #2 }
1705     \stex_debug:nn{modules}{Importing~module:~
1706         \l_stex_import_ns_str ? \l_stex_import_name_str
1707     }
1708     \stex_if_smsmode:F {
1709         \stex_import_require_module:nnnn
1710         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1711         { \l_stex_import_path_str } { \l_stex_import_name_str }
1712         \stex_annotate_invisible:nnn
1713         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1714     }
1715     \exp_args:Nx \stex_add_to_current_module:n {
1716         \stex_import_require_module:nnnn
1717         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1718         { \l_stex_import_path_str } { \l_stex_import_name_str }
1719     }
1720     \exp_args:Nx \stex_add_import_to_current_module:n {
1721         \l_stex_import_ns_str ? \l_stex_import_name_str
1722     }

```

```

1723 \stex_smsmode_set_codes:
1724 }
1725 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for \importmodule. This function is documented on page 31.)

\usemodule

```

1726 \NewDocumentCommand \usemodule { 0{} m } {
1727 \stex_if_smsmode:F {
1728 \stex_import_module_uri:nn { #1 } { #2 }
1729 \stex_import_require_module:nnnn
1730 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1731 { \l_stex_import_path_str } { \l_stex_import_name_str }
1732 \stex_annotate_invisible:nnn
1733 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1734 }
1735 \stex_smsmode_set_codes:
1736 }

```

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

```

1737 \endpackage

```

Chapter 30

STEX -Symbols Implementation

```
1738 <*package>
1739
1740 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1741
```

Warnings and error messages

```
1742
```

30.1 Symbol Declarations

```
1743 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1744 \seq_new:N \l_stex_all_symbols_seq
```

(End definition for `\l_stex_all_symbols_seq`. This variable is documented on page 35.)

`\STEXsymbol`

```
1745 \NewDocumentCommand \STEXsymbol { m } {
1746   \stex_get_symbol:n { #1 }
1747   \exp_args:No
1748   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1749 }
```

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

symdecl arguments:

```
1750 \keys_define:nn { stex / symdecl } {
1751   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1752   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1753   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1754   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1755   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1756   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1757   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1758   def       .tl_set:N = \l_stex_symdecl_definiens_tl
1759 }
```



```

1760
1761 \bool_new:N \l_stex_symdecl_make_macro_bool
1762
1763 \cs_new_protected:Nn \__stex_symdecl_args:n {
1764   \str_clear:N \l_stex_symdecl_name_str
1765   \str_clear:N \l_stex_symdecl_args_str
1766   \bool_set_false:N \l_stex_symdecl_local_bool
1767   \tl_clear:N \l_stex_symdecl_type_tl
1768   \tl_clear:N \l_stex_symdecl_definiens_tl
1769
1770   \keys_set:nn { stex / symdecl } { #1 }
1771 }

```

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

```

1772
1773 \NewDocumentCommand \symdecl { s O{} m } {
1774   \__stex_symdecl_args:n { #2 }
1775   \IfBooleanTF #1 {
1776     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1777   } {
1778     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1779   }
1780   \stex_symdecl_do:n { #3 }
1781   \stex_smsmode_set_codes:
1782 }
1783 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 34.)

\stex_symdecl_do:n

```

1784 \cs_new_protected:Nn \stex_symdecl_do:n {
1785   \stex_if_in_module:F {
1786     % TODO throw error? some default namespace?
1787   }
1788
1789   \str_if_empty:NT \l_stex_symdecl_name_str {
1790     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1791   }
1792
1793   \prop_if_exist:cT { l_stex_symdecl_
1794     \l_stex_current_module_str ?
1795     \l_stex_symdecl_name_str
1796   }_prop
1797 }{
1798   % TODO throw error (beware of circular dependencies)
1799 }
1800
1801 \prop_clear:N \l_tmpa_prop
1802 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1803 \seq_clear:N \l_tmpa_seq
1804 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1805 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1806

```

```

1807 \exp_args:No \stex_add_constant_to_current_module:n {
1808   \l_stex_symdecl_name_str
1809 }
1810
1811 % arity/args
1812 \int_zero:N \l_tmpb_int
1813
1814 \bool_set_true:N \l_tmpa_bool
1815 \str_map_inline:Nn \l_stex_symdecl_args_str {
1816   \token_case_meaning:NnF ##1 {
1817     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1818     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1819     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1820     {\tl_to_str:n a} {
1821       \bool_set_false:N \l_tmpa_bool
1822       \int_incr:N \l_tmpb_int
1823     }
1824     {\tl_to_str:n B} {
1825       \bool_set_false:N \l_tmpa_bool
1826       \int_incr:N \l_tmpb_int
1827     }
1828   }{
1829     \msg_set:nnn{stex}{error/wrongargs}{
1830       args~value~in~symbol~declaration~for~
1831       \l_stex_current_module_str ?
1832       \l_stex_symdecl_name_str ~
1833       needs~to~be~
1834       i,~a,~b~or~B,~but~##1~given
1835     }
1836     \msg_error:nn{stex}{error/wrongargs}
1837   }
1838 }
1839 \bool_if:NTF \l_tmpa_bool {
1840   % possibly numeric
1841   \str_if_empty:NTF \l_stex_symdecl_args_str {
1842     \prop_put:Nnn \l_tmpa_prop { args } {}
1843     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1844   }{
1845     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1846     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1847     \str_clear:N \l_tmpa_str
1848     \int_step_inline:nn \l_tmpa_int {
1849       \str_put_right:Nn \l_tmpa_str i
1850     }
1851     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1852   }
1853 } {
1854   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1855   \prop_put:Nnx \l_tmpa_prop { arity }
1856     { \str_count:N \l_stex_symdecl_args_str }
1857 }
1858 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1859
1860

```

```

1861 % semantic macro
1862
1863 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1864   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1865     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1866   } }
1867
1868   \bool_if:NF \l_stex_symdecl_local_bool {
1869     \exp_args:Nx \stex_add_to_current_module:n {
1870       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1871         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1872       } }
1873     }
1874   }
1875 }
1876
1877 % add to all symbols
1878
1879 \bool_if:NF \l_stex_symdecl_local_bool {
1880   \exp_args:Nx \stex_add_to_current_module:n {
1881     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1882       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1883     }
1884   }
1885   % \exp_args:Nx \stex_add_field_to_current_module:n {
1886   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1887   % }
1888 }
1889
1890 \stex_debug:nn{symbols}{New~symbol:~
1891   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1892   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1893   Args:~\prop_item:Nn \l_tmpa_prop { args }
1894 }
1895
1896 % circular dependencies require this:
1897
1898 \prop_if_exist:cF {
1899   l_stex_symdecl_
1900   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1901   _prop
1902 } {
1903   \prop_set_eq:cn {
1904     l_stex_symdecl_
1905     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1906     _prop
1907   } \l_tmpa_prop
1908 }
1909
1910 \seq_clear:c {
1911   l_stex_symdecl_
1912   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1913   _notations
1914 }

```

```

1915
1916 \bool_if:NF \l_stex_symdecl_local_bool {
1917   \exp_args:Nx
1918   \stex_add_to_current_module:n {
1919     \seq_clear:c {
1920       l_stex_symdecl_
1921       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1922       _notations
1923     }
1924     \prop_set_from_keyval:cn {
1925       l_stex_symdecl_
1926       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1927       _prop
1928     } {
1929       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1930       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1931       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1932       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1933       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1934       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1935     }
1936   }
1937 }
1938
1939 \stex_if_smsmode:TF {
1940   \bool_if:NF \l_stex_symdecl_local_bool {
1941     % \exp_args:Nx \stex_add_to_sms:n {
1942     %   \prop_set_from_keyval:cn {
1943     %     l_stex_symdecl_
1944     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1945     %     _prop
1946     %   } {
1947     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1948     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1949     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1950     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1951     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1952     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1953     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1954     %   }
1955     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1956     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1957     %   }
1958     % }
1959   }
1960 }{
1961   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1962     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1963   }
1964   \stex_if_do_html:T {
1965     \stex_annotate_invisible:nnn {symdecl} {
1966       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1967     } {
1968       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st

```

```

1969     \stex_annotate_invisible:nnn{args}{}{
1970     \prop_item:Nn \l_tmpa_prop { args }
1971   }
1972   \stex_annotate_invisible:nnn{macroname}{#1}{}
1973   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1974     \stex_annotate_invisible:nnn{definiens}{}
1975     { $\l_stex_symdecl_definiens_tl$ }
1976   }
1977 }
1978 }
1979 }
1980 }

```

(End definition for \stex_symdecl_do:n. This function is documented on page 35.)

\stex_get_symbol:n

```

1981 \str_new:N \l_stex_get_symbol_uri_str
1982
1983 \cs_new_protected:Nn \stex_get_symbol:n {
1984   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1985     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1986   }{
1987     % argument is a string
1988     % is it a command name?
1989     \cs_if_exist:cTF { #1 }{
1990       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1991       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1992       \str_if_empty:NTF \l_tmpa_str {
1993         \exp_args:Nx \cs_if_eq:NNTF {
1994           \tl_head:N \l_tmpa_tl
1995         } \stex_invoke_symbol:n {
1996           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1997         }{
1998           \__stex_symdecl_get_symbol_from_string:n { #1 }
1999         }
2000       } {
2001         \__stex_symdecl_get_symbol_from_string:n { #1 }
2002       }
2003     }{
2004       % argument is not a command name
2005       \__stex_symdecl_get_symbol_from_string:n { #1 }
2006       % \l_stex_all_symbols_seq
2007     }
2008   }
2009 }
2010
2011 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2012   \str_set:Nn \l_tmpa_str { #1 }
2013   \bool_set_false:N \l_tmpa_bool
2014   \stex_if_in_module:T {
2015     \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2016       \bool_set_true:N \l_tmpa_bool
2017       \str_set:Nx \l_stex_get_symbol_uri_str {
2018         \l_stex_current_module_str ? #1

```

```

2019     }
2020   }
2021 }
2022 \bool_if:NF \l_tmpa_bool {
2023   \tl_set:Nn \l_tmpa_tl {
2024     \msg_set:nnn{stex}{error/unknownsymbol}{
2025       No~symbol~#1~found!
2026     }
2027     \msg_error:nn{stex}{error/unknownsymbol}
2028   }
2029   \str_set:Nn \l_tmpa_str { #1 }
2030   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2031   \seq_map_inline:Nn \l_stex_all_symbols_seq {
2032     \str_set:Nn \l_tmpb_str { ##1 }
2033     \str_if_eq:eeT { \l_tmpa_str } {
2034       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2035     } {
2036       \seq_map_break:n {
2037         \tl_set:Nn \l_tmpa_tl {
2038           \str_set:Nn \l_stex_get_symbol_uri_str {
2039             ##1
2040           }
2041         }
2042       }
2043     }
2044   }
2045   \l_tmpa_tl
2046 }
2047 }
2048
2049 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2050   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2051   { \tl_tail:N \l_tmpa_tl }
2052   \tl_if_single:NTF \l_tmpa_tl {
2053     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2054       \exp_after:wN \str_set:Nn \exp_after:wN
2055       \l_stex_get_symbol_uri_str \l_tmpa_tl
2056     }{
2057       % TODO
2058       % tail is not a single group
2059     }
2060   }{
2061     % TODO
2062     % tail is not a single group
2063   }
2064 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page [35](#).)

30.2 Notations

```

2065 <@@=stex_notation>

```

notation arguments:

```

2066 \keys_define:nn { stex / notation } {
2067   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2068   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2069   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2070   op        .tl_set:N   = \l__stex_notation_op_tl ,
2071   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2072   primary   .default:n  = {true} ,
2073   unknown   .code:n     = \str_set:Nx
2074     \l__stex_notation_variant_str \l_keys_key_str
2075 }
2076
2077 \cs_new_protected:Nn \stex_notation_args:n {
2078   \str_clear:N \l__stex_notation_lang_str
2079   \str_clear:N \l__stex_notation_variant_str
2080   \str_clear:N \l__stex_notation_prec_str
2081   \tl_clear:N \l__stex_notation_op_tl
2082   \bool_set_false:N \l__stex_notation_primary_bool
2083
2084   \keys_set:nn { stex / notation } { #1 }
2085 }

```

\notation

```

2086 \NewDocumentCommand \notation { 0{ } m } {
2087   \stex_notation_args:n { #1 }
2088   \tl_clear:N \l_stex_symdecl_definiens_tl
2089   \stex_get_symbol:n { #2 }
2090   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2091 }
2092 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2093 \cs_new_protected:Nn \stex_notation_do:nn {
2094   \let\l_stex_current_symbol_str\relax
2095   \prop_set_eq:Nc \l_tmpa_prop {
2096     l_stex_symdecl_ #1 _prop
2097   }
2098
2099   \prop_clear:N \l_tmpb_prop
2100   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2101   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2102   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2103
2104   % precedences
2105   \seq_clear:N \l_tmpb_seq
2106   \exp_args:NNno
2107   \str_if_empty:NTF \l__stex_notation_prec_str {
2108     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2109     \int_compare:nNnTF \l_tmpa_str = 0 {
2110       \exp_args:NNnx
2111       \prop_put:Nno \l_tmpb_prop { opprec }
2112       { \neginfprec }
2113     }{
2114       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }

```

```

2115     }
2116   } {
2117     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2118       \exp_args:NNnx
2119       \prop_put:Nno \l_tmpb_prop { opprec }
2120       { \neginfprec }
2121       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2122       \int_step_inline:nn { \l_tmpa_str } {
2123         \exp_args:NNx
2124         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2125       }
2126     }{
2127       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2128       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2129         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2130         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2131           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2132             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2133           \seq_map_inline:Nn \l_tmpa_seq {
2134             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2135           }
2136         }
2137         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2138       }{
2139         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2140         \int_compare:nNnTF \l_tmpa_str = 0 {
2141           \exp_args:NNnx
2142           \prop_put:Nno \l_tmpb_prop { opprec }
2143           { \infprec }
2144         }{
2145           \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2146         }
2147       }
2148     }
2149   }
2150
2151   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2152   \int_step_inline:nn { \l_tmpa_str } {
2153     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2154       \exp_args:NNx
2155       \seq_put_right:Nn \l_tmpb_seq {
2156         \prop_item:Nn \l_tmpb_prop { opprec }
2157       }
2158     }
2159   }
2160
2161   \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2162   \tl_clear:N \l_tmpa_tl
2163
2164   \int_compare:nNnTF \l_tmpa_str = 0 {
2165     \exp_args:NNe
2166     \cs_set:Npn \l__stex_notation_macrocode_cs {
2167       \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2168       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```



```

2169         { \prop_item:Nn \l_tmpb_prop { opprec } }
2170         { \exp_not:n { #2 } }
2171     }
2172     \__stex_notation_final:
2173 }{
2174     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2175     \str_if_in:NnTF \l_tmpb_str b {
2176         \exp_args:Nne \use:nn
2177         {
2178             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2179             \cs_set:Npn \l_tmpa_str } { {
2180                 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2181                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2182                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2183                 { \exp_not:n { #2 } }
2184             } }
2185         }{
2186             \str_if_in:NnTF \l_tmpb_str B {
2187                 \exp_args:Nne \use:nn
2188                 {
2189                     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2190                     \cs_set:Npn \l_tmpa_str } { {
2191                         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2192                         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2193                         { \prop_item:Nn \l_tmpb_prop { opprec } }
2194                         { \exp_not:n { #2 } }
2195                     } }
2196                 }{
2197                     \exp_args:Nne \use:nn
2198                     {
2199                         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2200                         \cs_set:Npn \l_tmpa_str } { {
2201                             \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2202                             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2203                             { \prop_item:Nn \l_tmpb_prop { opprec } }
2204                             { \exp_not:n { #2 } }
2205                         } }
2206                     }
2207                 }
2208
2209                 \int_zero:N \l_tmpa_int
2210                 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2211                 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2212                 \__stex_notation_arguments:
2213             }
2214         }

```

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

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2215 \cs_new_protected:Nn \__stex_notation_arguments: {
2216     \int_incr:N \l_tmpa_int
2217     \str_if_empty:NnTF \l_tmpa_str {
2218         \__stex_notation_final:

```

```

2219 }{
2220   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2221   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2222   \str_if_eq:VnTF \l_tmpb_str a {
2223     \__stex_notation_argument_assoc:n
2224   }{
2225     \str_if_eq:VnTF \l_tmpb_str B {
2226       \__stex_notation_argument_assoc:n
2227     }{
2228       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2229       \tl_put_right:Nx \l_tmpa_tl {
2230         { \stex_term_math_arg:nnn
2231           { \int_use:N \l_tmpa_int }
2232           { \l_tmpb_str }
2233           { ####\int_use:N \l_tmpa_int }
2234         }
2235       }
2236       \__stex_notation_arguments:
2237     }
2238   }
2239 }
2240 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2241 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2242   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2243   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2244   \tl_put_right:Nx \l_tmpa_tl {
2245     { \stex_term_math_assoc_arg:nnnn
2246       { \int_use:N \l_tmpa_int }
2247       { \l_tmpb_str }
2248       \exp_args:No \exp_not:n
2249       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2250       { ####\int_use:N \l_tmpa_int }
2251     }
2252   }
2253   \__stex_notation_arguments:
2254 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2255 \cs_new_protected:Nn \__stex_notation_final: {
2256   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2257   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2258   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2259   \exp_args:Nne \use:nn
2260   {
2261     \cs_generate_from_arg_count:cNnn {
2262       stex_notation_ \l_tmpa_str \c_hash_str
2263       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2264       _cs

```

```

2265 }
2266 \cs_set:Npn \l_tmpb_str } { {
2267   \exp_after:wN \exp_after:wN \exp_after:wN
2268   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2269   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2270 } }
2271
2272 \tl_if_empty:NF \l__stex_notation_op_tl {
2273   \cs_set:cpx {
2274     stex_op_notation_ \l_tmpa_str \c_hash_str
2275     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2276     _cs
2277   } {
2278     \stex_term_oms:nnn {
2279       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2280       \l__stex_notation_lang_str
2281     }{
2282       \l_tmpa_str
2283     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2284   }
2285 }
2286
2287 \exp_args:Ne
2288 \stex_add_to_current_module:n {
2289   \cs_generate_from_arg_count:cNnn {
2290     stex_notation_ \l_tmpa_str \c_hash_str
2291     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2292     _cs
2293   } \cs_set:Npn {\l_tmpb_str} {
2294     \exp_after:wN \exp_after:wN \exp_after:wN
2295     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2296     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2297   }
2298   \tl_if_empty:NF \l__stex_notation_op_tl {
2299     \cs_set:cpn {
2300       stex_op_notation_ \l_tmpa_str \c_hash_str
2301       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2302       _cs
2303     } {
2304       \stex_term_oms:nnn {
2305         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2306         \l__stex_notation_lang_str
2307       }{
2308         \l_tmpa_str
2309       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2310     }
2311   }
2312 }
2313
2314 \seq_put_right:cx {
2315   l_stex_symdecl_
2316   \prop_item:Nn \l_tmpb_prop { symbol }
2317   _notations
2318 } {

```

```

2319 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2320 }
2321
2322 \stex_debug:nn{symbols}{
2323   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2324   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2325   Operator~precedence:~
2326   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2327   Argument~precedences:~
2328   \seq_use:Nn \l_tmpa_seq {,~}^^J
2329   Notation: \cs_meaning:c {
2330     stex_notation_ \l_tmpa_str \c_hash_str
2331     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2332     _cs
2333   }
2334 }
2335
2336 \prop_set_eq:cN {
2337   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2338   \c_hash_str \l__stex_notation_lang_str _prop
2339 } \l_tmpb_prop
2340
2341 \exp_args:Ne
2342 \stex_add_to_current_module:n {
2343   \seq_put_right:cn {
2344     l_stex_symdecl_
2345     \prop_item:Nn \l_tmpb_prop { symbol }
2346     _notations
2347   } {
2348     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2349   }
2350   \prop_set_from_keyval:cn {
2351     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2352     \c_hash_str \l__stex_notation_lang_str _prop
2353   } {
2354     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2355     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2356     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2357     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2358     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2359   }
2360 }
2361
2362 \stex_if_smsmode:TF {
2363   \stex_smsmode_set_codes:
2364   % \exp_args:Nx \stex_add_to_sms:n {
2365   %   \prop_set_from_keyval:cn {
2366   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2367   %     \c_hash_str \l__stex_notation_lang_str _prop
2368   %   } {
2369   %     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2370   %     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2371   %     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2372   %     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,

```

```

2373 %      argprec = \prop_item:Nn \l_tmpb_prop { argprec } ,
2374 %    }
2375 %  }
2376 }{
2377
2378 % HTML annotations
2379 \stex_if_do_html:T {
2380   \stex_annotate_invisible:nnn { notation }
2381   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2382     \stex_annotate_invisible:nnn { notationfragment }
2383     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2384     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2385     \stex_annotate_invisible:nnn { precedence }
2386     { \prop_item:Nn \l_tmpb_prop { opprec };
2387       \seq_use:Nn \l_tmpa_seq { x }
2388     }{}
2389
2390     \int_zero:N \l_tmpa_int
2391     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2392     \tl_clear:N \l_tmpa_tl
2393     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2394       \int_incr:N \l_tmpa_int
2395       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2396       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2397       \str_if_eq:VnTF \l_tmpb_str a {
2398         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2399           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2400           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2401         } }
2402       }{
2403         \str_if_eq:VnTF \l_tmpb_str B {
2404           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2405             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2406             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2407           } }
2408         }{
2409           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2410             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2411           } }
2412         }
2413       }
2414     }
2415     \stex_annotate_invisible:nnn { notationcomp }{}{
2416       \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2417       $ \exp_args:Nno \use:nn { \use:c {
2418         stex_notation_ \l_stex_current_symbol_str
2419         \c_hash_str \l__stex_notation_variant_str
2420         \c_hash_str \l__stex_notation_lang_str _cs
2421       } } { \l_tmpa_tl } $
2422     }
2423   }
2424 }
2425 }
2426 }

```

(End definition for `_stex_notation_final:`)

`\symdef`

```

2427 \keys_define:nn { stex / symdef } {
2428   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2429   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2430   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2431   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2432   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2433   op        .tl_set:N = \l__stex_notation_op_tl ,
2434   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2435   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2436   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2437   unknown   .code:n = \str_set:Nx
2438             \l__stex_notation_variant_str \l_keys_key_str
2439 }
2440
2441 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2442   \str_clear:N \l_stex_symdecl_name_str
2443   \str_clear:N \l_stex_symdecl_args_str
2444   \bool_set_false:N \l_stex_symdecl_local_bool
2445   \tl_clear:N \l_stex_symdecl_type_tl
2446   \tl_clear:N \l_stex_symdecl_definiens_tl
2447   \str_clear:N \l__stex_notation_lang_str
2448   \str_clear:N \l__stex_notation_variant_str
2449   \str_clear:N \l__stex_notation_prec_str
2450   \tl_clear:N \l__stex_notation_op_tl
2451
2452   \keys_set:nn { stex / symdef } { #1 }
2453 }
2454
2455 \NewDocumentCommand \symdef { 0{} m } {
2456   \_stex_notation_symdef_args:n { #1 }
2457   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2458   \stex_symdecl_do:n { #2 }
2459   \exp_args:Nx \stex_notation_do:nn {
2460     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2461   }
2462 }
2463 \stex_deactivate_macro:Nn \symdef {module~environments}
2464
2465 (End definition for \symdef. This function is documented on page 36.)
2466 \endpackage

```

Chapter 31

STEX -Terms Implementation

```
2465 <*package>
2466
2467 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2468
2469 <@@=stex_terms>
2470
2471 Warnings and error messages
2472 \msg_new:nnn{stex}{error/nonotation}{
2473   Symbol~#1~invoked,~but~has~no~notation~#2!
2474 }
2475 \msg_new:nnn{stex}{error/notationarg}{
2476   Error~in~parsing~notation~#1
2477 }
2478 \msg_new:nnn{stex}{error/noop}{
2479   Symbol~#1~has~no~operator~notation~for~notation~#2
2480 }
```

31.1 Symbol Invocations

Arguments:

```
2480 \keys_define:nn { stex / terms } {
2481   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2482   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2483   unknown .code:n = \str_set:Nx
2484     \l__stex_terms_variant_str \l_keys_key_str
2485 }
2486
2487 \cs_new_protected:Nn \__stex_terms_args:n {
2488   \str_clear:N \l__stex_terms_lang_str
2489   \str_clear:N \l__stex_terms_variant_str
2490   \str_clear:N \l__stex_terms_prec_str
2491   \tl_clear:N \l__stex_terms_op_tl
2492 }
2493 \keys_set:nn { stex / terms } { #1 }
```

2494 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2495 \cs_new_protected:Nn \stex_invoke_symbol:n {
2496   \if_mode_math:
2497     \exp_after:wN \__stex_terms_invoke_math:n
2498   \else:
2499     \exp_after:wN \__stex_terms_invoke_text:n
2500   \fi: { #1 }
2501 }
```

(End definition for \stex_invoke_symbol:n. This function is documented on page 37.)

__stex_terms_invoke_math:n

```
2502 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2503   \peek_charcode_remove:NTF ! {
2504     \peek_charcode:NTF [ {
2505       \__stex_terms_invoke_op:nw { #1 }
2506     }{
2507       \peek_charcode_remove:NTF ! {
2508         \peek_charcode:NTF [ {
2509           \__stex_terms_invoke_op_custom:nw
2510         }{
2511           % TODO throw error
2512         }
2513       }{
2514         \__stex_terms_invoke_op:nw { #1 } []
2515       }
2516     }
2517   }{
2518     \peek_charcode_remove:NTF * {
2519       \__stex_terms_invoke_text:n { #1 }
2520     }{
2521       \peek_charcode:NTF [ {
2522         \__stex_terms_invoke_math:nw { #1 }
2523       }{
2524         \__stex_terms_invoke_math:nw { #1 } []
2525       }
2526     }
2527   }
2528 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2529 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2530   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2531     \stex_highlight_term:nn{#1}{#2}
2532   }
2533 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2534 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2535   \_stex_terms_args:n { #2 }
2536   \cs_if_exist:cTF {
2537     stex_op_notation_ #1 \c_hash_str
2538     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2539   }{
2540     \csname stex_op_notation_ #1 \c_hash_str
2541       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2542     \endcsname
2543   }{
2544     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2545   }
2546 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2547 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2548   \_stex_terms_args:n { #2 }
2549   \seq_if_empty:cTF {
2550     l_stex_symdecl_ #1 _notations
2551   } {
2552     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2553   } {
2554     \seq_if_in:cxTF {
2555       l_stex_symdecl_ #1 _notations
2556     }
2557     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2558       \str_set:Nn \l_stex_current_symbol_str { #1 }
2559       \use:c{
2560         stex_notation_ #1 \c_hash_str
2561         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2562         _cs
2563       }
2564     }{
2565       \str_if_empty:NTF \l__stex_terms_variant_str {
2566         \str_if_empty:NTF \l__stex_terms_lang_str {
2567           \seq_get_left:cN {
2568             l_stex_symdecl_ #1 _notations
2569           } \l_tmpa_str
2570           \str_set:Nn \l_stex_current_symbol_str { #1 }
2571           \use:c{
2572             stex_notation_ #1 \c_hash_str \l_tmpa_str
2573             _cs
2574           }
2575         }{
2576           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2577             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2578           }
2579         }
2580       }{
2581         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2582           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2583     }
2584   }
2585 }
2586 }
2587 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2588 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2589   \peek_charcode_remove:NTF ! {
2590     \stex_term_custom:nn { #1 } { }
2591   }{
2592     \prop_set_eq:Nc \l_tmpa_prop {
2593       l_stex_symdecl_ #1 _prop
2594     }
2595     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2596     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2597   }
2598 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2599 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2600 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2601 \int_new:N \l__stex_terms_downprec
2602 \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 38.)

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2603 \tl_set:Nn \l__stex_terms_left_bracket_str (
2604 \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

```

2605 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2606   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2607     \bool_set_false:N \l__stex_terms_brackets_done_bool
2608     #2
2609   } {
2610     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2611       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2612         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2613         \dobrackets { #2 }
2614       }

```

```

2615     }{ #2 }
2616   }
2617 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2618 \bool_new:N \l__stex_terms_brackets_done_bool
2619 %\RequirePackage{scalerel}
2620 \cs_new_protected:Npn \dobrackets #1 {
2621   %\ThisStyle{\if D\m@switch
2622   %   \exp_args:Nnx \use:nn
2623   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2624   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2625   % \else
2626   %   \exp_args:Nnx \use:nn
2627   %   {
2628   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
2629   %     \int_set:Nn \l__stex_terms_downprec \infprec
2630   %     \l__stex_terms_left_bracket_str
2631   %     #1
2632   %   }
2633   %   {
2634   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
2635   %     \l__stex_terms_right_bracket_str
2636   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2637   %   }
2638   %\fi}
2639 }

```

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

`\withbrackets`

```

2640 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2641   \exp_args:Nnx \use:nn
2642   {
2643     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2644     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2645     #3
2646   }
2647   {
2648     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2649     {\l__stex_terms_left_bracket_str}
2650     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2651     {\l__stex_terms_right_bracket_str}
2652   }
2653 }

```

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

`\STEXinvisible`

```

2654 \cs_new_protected:Npn \STEXinvisible #1 {
2655   \stex_annotate_invisible:n { #1 }
2656 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2657 \cs_new_protected:Nn \_stex_term_oms:nnn {
2658   \stex_annotate:nnn{ OMID }{ #2 }{
2659     \stex_highlight_term:nn { #1 } { #3 }
2660   }
2661 }
2662
2663 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2664   \__stex_terms_maybe_brackets:nn { #3 }{
2665     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2666   }
2667 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 37.)

`_stex_term_math_oma:nnnn`

```

2668 \cs_new_protected:Nn \_stex_term_oma:nnn {
2669   \stex_annotate:nnn{ OMA }{ #2 }{
2670     \stex_highlight_term:nn { #1 } { #3 }
2671   }
2672 }
2673
2674 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2675   \__stex_terms_maybe_brackets:nn { #3 }{
2676     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2677   }
2678 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 37.)

`_stex_term_math_omb:nnnn`

```

2679 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2680   \stex_annotate:nnn{ OMBIND }{ #2 }{
2681     \stex_highlight_term:nn { #1 } { #3 }
2682   }
2683 }
2684
2685 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2686   \__stex_terms_maybe_brackets:nn { #3 }{
2687     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2688   }
2689 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 37.)

`_stex_term_math_arg:nnn`

```

2690 \cs_new_protected:Nn \_stex_term_arg:nn {
2691   \stex_unhighlight_term:n {
2692     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2693   }
2694 }

```

```

2695 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2696   \exp_args:Nnx \use:nn
2697     { \int_set:Nn \l__stex_terms_downprec { #2 }
2698       \stex_term_arg:nn { #1 }{ #3 }
2699     }
2700   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2701   }

```

(End definition for `\stex_term_math_arg:nnn`. This function is documented on page 37.)

`\stex_term_math_assoc_arg:nnnn`

```

2702 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2703   \clist_set:Nn \l_tmpa_clist{ #4 }
2704   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2705     \tl_set:Nn \l_tmpa_tl { #4 }
2706   }{
2707     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2708     \clist_reverse:N \l_tmpa_clist
2709     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2710
2711     \clist_map_inline:Nn \l_tmpa_clist {
2712       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2713         \exp_args:Nno
2714           \l_tmpa_cs { ##1 } \l_tmpa_tl
2715       }
2716     }
2717
2718   }
2719   \exp_args:Nnno
2720   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2721 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 37.)

`\stex_term_custom:nn`

```

2722 \cs_new_protected:Nn \stex_term_custom:nn {
2723   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2724   \str_set:Nn \l_tmpa_str { #2 }
2725   \tl_clear:N \l_tmpa_tl
2726   \int_zero:N \l_tmpa_int
2727   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2728   \__stex_terms_custom_loop:
2729 }

```

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

`__stex_terms_custom_loop:`

```

2730 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2731   \bool_set_false:N \l_tmpa_bool
2732   \bool_while_do:nn {
2733     \str_if_eq_p:ee X {
2734       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2735     }
2736   }{
2737     \int_incr:N \l_tmpa_int

```

```

2738 }
2739
2740 \peek_charcode:NTF [ {
2741   % notation/text component
2742   \__stex_terms_custom_component:w
2743 } {
2744   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2745     % all arguments read => finish
2746     \__stex_terms_custom_final:
2747   } {
2748     % arguments missing
2749     \peek_charcode_remove:NTF * {
2750       % invisible, specific argument position or both
2751       \peek_charcode:NTF [ {
2752         % visible specific argument position
2753         \__stex_terms_custom_arg:wn
2754       } {
2755         % invisible
2756         \peek_charcode_remove:NTF * {
2757           % invisible specific argument position
2758           \__stex_terms_custom_arg_inv:wn
2759         } {
2760           % invisible next argument
2761           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2762         }
2763       }
2764     } {
2765       % next normal argument
2766       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2767     }
2768   }
2769 }
2770 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2771 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2772   \bool_set_true:N \l_tmpa_bool
2773   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2774 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2775 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2776   \str_set:Nx \l_tmpb_str {
2777     \str_item:Nn \l_tmpa_str { #1 }
2778   }
2779   \str_case:VnTF \l_tmpb_str {
2780     { X } {
2781       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2782     }
2783     { i } { \__stex_terms_custom_set_X:n { #1 } }
2784     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2785 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2786 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2787 }{}{
2788 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2789 }
2790
2791 \bool_if:nTF \l_tmpa_bool {
2792   \tl_put_right:Nx \l_tmpa_tl {
2793     \stex_annotate_invisible:n {
2794       \stex_term_arg:nn { \int_eval:n { #1 } }
2795       \exp_not:n { { #2 } }
2796     }
2797   }
2798 } {
2799   \tl_put_right:Nx \l_tmpa_tl {
2800     \stex_term_arg:nn { \int_eval:n { #1 } }
2801     \exp_not:n { { #2 } }
2802   }
2803 }
2804
2805 \_stex_terms_custom_loop:
2806 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2807 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2808   \str_set:Nx \l_tmpa_str {
2809     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2810     X
2811     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2812   }
2813 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2814 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2815   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2816   \_stex_terms_custom_loop:
2817 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2818 \cs_new_protected:Nn \_stex_terms_custom_final: {
2819   \int_compare:nNnTF \l_tmpb_int = 0 {
2820     \exp_args:Nnno \stex_term_oms:nnn
2821   }{
2822     \str_if_in:NnTF \l_tmpa_str {b} {
2823       \exp_args:Nnno \stex_term_ombind:nnn
2824     } {
2825       \exp_args:Nnno \stex_term_oma:nnn
2826     }
2827   }

```

```

2828 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2829 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

2830 \NewDocumentCommand \symref { m m }{
2831   \let\compemph_uri_prev:\compemph@uri
2832   \let\compemph@uri\symrefemph@uri
2833   \STEXsymbol{#1}! [#2]
2834   \let\compemph@uri\compemph_uri_prev:
2835 }
2836
2837 \keys_define:nn { stex / symname } {
2838   post      .str_set_x:N   = \l_stex_symname_post_str
2839 }
2840
2841 \cs_new_protected:Nn \stex_symname_args:n {
2842   \str_clear:N \l_stex_symname_post_str
2843   \keys_set:nn { stex / symname } { #1 }
2844 }
2845
2846 \NewDocumentCommand \symname { 0{} m }{
2847   \stex_symname_args:n { #1 }
2848   \stex_get_symbol:n { #2 }
2849   \str_set:Nx \l_tmpa_str {
2850     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2851   }
2852   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2853
2854   \let\compemph_uri_prev:\compemph@uri
2855   \let\compemph@uri\symrefemph@uri
2856   \exp_args:NNx \use:nn
2857   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2858     \l_tmpa_str \l_stex_symname_post_str
2859   ] }
2860   \let\compemph@uri\compemph_uri_prev:
2861 }

```

(End definition for \symref and \symname. These functions are documented on page 37.)

31.3 Notation Components

```

2862 <@@=stex_notationcomps>

```

\stex_highlight_term:nn

```

2863
2864 \str_new:N \l_stex_current_symbol_str
2865 \cs_new_protected:Nn \stex_highlight_term:nn {
2866   \exp_args:Nnx
2867   \use:nn {
2868     \str_set:Nx \l_stex_current_symbol_str { #1 }
2869     #2
2870   } {

```



```

2871 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2872 { \l_stex_current_symbol_str }
2873 }
2874 }
2875
2876 \cs_new_protected:Nn \stex_unhighlight_term:n {
2877 % \latexml_if:TF {
2878 % #1
2879 % } {
2880 % \rustex_if:TF {
2881 % #1
2882 % } {
2883 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2884 % }
2885 % }
2886 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2887 \cs_new_protected:Npn \comp #1 {
2888 \str_if_empty:NF \l_stex_current_symbol_str {
2889 \rustex_if:TF {
2890 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
2891 }{
2892 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2893 }
2894 }
2895 }
2896
2897 \cs_new_protected:Npn \compemph@uri #1 #2 {
2898 \compemph{ #1 }
2899 }
2900
2901
2902 \cs_new_protected:Npn \compemph #1 {
2903 #1
2904 }
2905
2906 \cs_new_protected:Npn \defemph@uri #1 #2 {
2907 \defemph{#1}
2908 }
2909
2910 \cs_new_protected:Npn \defemph #1 {
2911 \textbf{#1}
2912 }
2913
2914 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2915 \symrefemph{#1}
2916 }
2917
2918 \cs_new_protected:Npn \symrefemph #1 {
2919 \textbf{#1}
2920 }

```

(End definition for `\comp` and others. These functions are documented on page 39.)

`\ellipses`

```
2921 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2922 \bool_new:N \l_stex_inarray_bool
\parrayline 2923 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2924 \NewDocumentCommand \parray { m m } {
\parraycell 2925 \begin{group}
2926 \bool_set_true:N \l_stex_inarray_bool
2927 \begin{array}{#1}
2928 #2
2929 \end{array}
2930 \end{group}
2931 }
2932
2933 \NewDocumentCommand \prmatrix { m } {
2934 \begin{group}
2935 \bool_set_true:N \l_stex_inarray_bool
2936 \begin{matrix}
2937 #1
2938 \end{matrix}
2939 \end{group}
2940 }
2941
2942 \def \maybepline {
2943 \bool_if:NT \l_stex_inarray_bool {\hline}
2944 }
2945
2946 \def \parrayline #1 #2 {
2947 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2948 }
2949
2950 \def \pmrow #1 { \parrayline{}{ #1 } }
2951
2952 \def \parraylineh #1 #2 {
2953 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2954 }
2955
2956 \def \parraycell #1 {
2957 #1 \bool_if:NT \l_stex_inarray_bool {&}
2958 }

```

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

```
2959 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
2960 <*package>
2961
2962 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2963
2964 <@@=stex_features>
    Warnings and error messages
2965
```

32.1 Imports with modification

```
2966 \cs_new_protected:Nn \stex_get_symbol_in_clonemodule:n {
2967   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2968     \__stex_features_get_symbol_from_cs:n { #1 }
2969   }{
2970     % argument is a string
2971     % is it a command name?
2972     \cs_if_exist:cTF { #1 }{
2973       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2974       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2975       \str_if_empty:NNTF \l_tmpa_str {
2976         \exp_args:Nx \cs_if_eq:NNTF {
2977           \tl_head:N \l_tmpa_tl
2978         } \stex_invoke_symbol:n {
2979           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
2980         }{
2981           \__stex_features_get_symbol_from_string:n { #1 }
2982         }
2983       } {
2984         \__stex_features_get_symbol_from_string:n { #1 }
2985       }
2986     }{
2987       % argument is not a command name
```

```

2988     \__stex_features_get_symbol_from_string:n { #1 }
2989     % \l_stex_all_symbols_seq
2990   }
2991 }
2992 }
2993
2994 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
2995   \str_set:Nn \l_tmpa_str { #1 }
2996   \bool_set_false:N \l_tmpa_bool
2997   \bool_if:NF \l_tmpa_bool {
2998     \tl_set:Nn \l_tmpa_tl {
2999       \msg_set:nnn{stex}{error/unknownsymbol}{
3000         No~symbol~#1~found!
3001       }
3002       \msg_error:nn{stex}{error/unknownsymbol}
3003     }
3004     \str_set:Nn \l_tmpa_str { #1 }
3005     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3006     \seq_map_inline:Nn \l__stex_features_clonemodule_fields_seq {
3007       \str_set:Nn \l_tmpb_str { ##1 }
3008       \str_if_eq:eeT { \l_tmpa_str } {
3009         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3010       } {
3011         \seq_map_break:n {
3012           \tl_set:Nn \l_tmpa_tl {
3013             \str_set:Nn \l_stex_get_symbol_uri_str {
3014               ##1
3015             }
3016             \__stex_features_get_symbol_check:
3017           }
3018         }
3019       }
3020     }
3021     \l_tmpa_tl
3022   }
3023 }
3024
3025 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3026   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3027   { \tl_tail:N \l_tmpa_tl }
3028   \tl_if_single:NTF \l_tmpa_tl {
3029     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3030       \exp_after:wN \str_set:Nn \exp_after:wN
3031       \l_stex_get_symbol_uri_str \l_tmpa_tl
3032       \__stex_features_get_symbol_check:
3033     }{
3034       % TODO
3035       % tail is not a single group
3036     }
3037   }{
3038     % TODO
3039     % tail is not a single group
3040   }
3041 }

```

```

3042
3043 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3044   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3045   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3046     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3047     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3048     \seq_if_in:NoF \l__stex_features_clonemodule_modules_seq {
3049       % TODO error
3050     }
3051   }{
3052     % TODO error
3053   }
3054 }
3055
3056 \NewDocumentEnvironment {clonemodule} { 0{} m m}{
3057   \stex_import_module_uri:nn { #1 } { #2 }
3058   \stex_deactivate_macro:Nn \symdecl {module~environments}
3059   \stex_deactivate_macro:Nn \symdef {module~environments}
3060   \stex_reactivate_macro:N \assign
3061   \stex_reactivate_macro:N \renamedec1
3062   \stex_reactivate_macro:N \donotclone
3063   \let\notation\notation_in_clonemodules:
3064   %\stex_module_setup:nn {}{ #3 }
3065   \stex_import_require_module:nnnn
3066     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3067     { \l_stex_import_path_str } { \l_stex_import_name_str }
3068   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3069   \seq_set_eq:NN \l__stex_features_clonemodule_modules_seq \l_stex_collect_imports_seq
3070   \seq_clear:N \l__stex_features_clonemodule_fields_seq
3071   \seq_map_inline:Nn \l__stex_features_clonemodule_modules_seq {
3072     \seq_map_inline:cn {c_stex_module_###1_constants}{
3073       \exp_args:NNx \seq_put_right:Nn \l__stex_features_clonemodule_fields_seq {
3074         ##1 ? #####1
3075       }
3076     }
3077   }
3078   \stex_debug:nn{clonemodule}{cloning~module~{\l_stex_import_ns_str ?\l_stex_import_name_str
3079     as~\l_stex_current_module_str?#3}
3080   \stex_debug:nn{clonemodule}{fields:\seq_use:Nn \l__stex_features_clonemodule_fields_seq {,
3081     % todo
3082   }{
3083     % todo
3084   }
3085
3086 \NewDocumentCommand \donotclone { 0{} m}{
3087   \stex_import_module_uri:nn { #1 } { #2 }
3088   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3089   % TODO add to structure somehow
3090   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3091     \seq_remove_all:Nn \l__stex_features_clonemodule_modules_seq { ##1 }
3092     \seq_map_inline:cn {c_stex_module_###1_constants}{
3093       \seq_remove_all:Nn \l__stex_features_clonemodule_fields_seq { ##1 ? #####1 }
3094       \bool_lazy_any_p:nT {
3095         { \cs_if_exist_p:c {l__stex_features_clonemodule_###1?#####1_name_str}}

```

```

3096         { \cs_if_exist_p:c {l__stex_features_clonemodule_##1?####1_macroname_str}}
3097         { \cs_if_exist_p:c {l__stex_features_clonemodule_##1?####1_def_tl}}
3098     }{
3099         % TODO throw error
3100     }
3101 }
3102 }
3103 }
3104
3105 \NewDocumentCommand \assign { m m }{
3106     \stex_get_symbol_in_clonemodule:n {#1}
3107     \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3108     \tl_set:cn {l__stex_features_clonemodule_##1?####1_def_tl}{#2}
3109 }
3110
3111 \keys_define:nn { stex / renamedec1 } {
3112     name .str_set_x:N = \l_stex_renamedec1_name_str
3113 }
3114 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3115     \str_clear:N \l_stex_renamedec1_name_str
3116
3117     \keys_set:nn { stex / renamedec1 } { #1 }
3118 }
3119
3120 \NewDocumentCommand \renamedec1 { 0{} m m }{
3121     \__stex_features_renamedec1_args:n { #1 }
3122     \stex_get_symbol_in_clonemodule:n {#2}
3123     \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3124     \str_set:cx {l__stex_features_clonemodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3125     \str_if_empty:NTF \l_stex_renamedec1_name_str {
3126         \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3127             \l_stex_get_symbol_uri_str
3128         } }
3129     } {
3130         \str_set:cx {l__stex_features_clonemodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_
3131         \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3132         \prop_set_eq:cc {l_stex_symdecl_
3133             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3134             _prop
3135         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3136         \seq_set_eq:cc {l_stex_symdecl_
3137             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3138             _notations
3139         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3140         \prop_put:cnx {l_stex_symdecl_
3141             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3142             _prop
3143         }{ name }{ \l_stex_renamedec1_name_str }
3144         \prop_put:cnx {l_stex_symdecl_
3145             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3146             _prop
3147         }{ module }{ \l_stex_current_module_str }
3148         \exp_args:NNx \seq_put_left:Nn \__stex_features_clonemodule_fields_seq {
3149             \l_stex_current_module_str ? \l_stex_renamedec1_name_str

```

```

3150     }
3151     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3152       \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3153     } }
3154   }
3155 }
3156 \NewDocumentCommand \notation_in_clonemodules: { 0{} m } {
3157   \_stex_notation_args:n { #1 }
3158   \tl_clear:N \l_stex_symdecl_definiens_tl
3159   \stex_get_symbol_in_clonemodule:n { #2 }
3160   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3161   % todo
3162 }
3163 \stex_deactivate_macro:Nn \assign {clonemodules}
3164 \stex_deactivate_macro:Nn \renameddecl {clonemodules}
3165 \stex_deactivate_macro:Nn \donotclone {clonemodules}
3166
3167
3168 \seq_new:N \l_stex_implicit_morphisms_seq
3169 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3170   \stex_import_module_uri:nn { #1 } { #2 }
3171   \stex_debug:nn{implicits}{
3172     Implicit~morphism:~
3173     \l_stex_module_ns_str ? \l__stex_features_name_str
3174   }
3175   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3176     \l_stex_module_ns_str ? \l__stex_features_name_str
3177   }{
3178     \msg_error:nnn{stex}{error/conflictingmodules}{
3179       \l_stex_module_ns_str ? \l__stex_features_name_str
3180     }
3181   }
3182
3183   % TODO
3184
3185
3186
3187   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3188     \l_stex_module_ns_str ? \l__stex_features_name_str
3189   }
3190 }
3191

```

32.2 The feature environment

structural@feature

```

3192
3193 \NewDocumentEnvironment{structural@feature}{ m m m }{
3194   \stex_if_in_module:F {
3195     \msg_set:nnn{stex}{error/nomodule}{
3196       Structural~Feature~has~to~occur~in~a~module:\\
3197       Feature~#2~of~type~#1\\
3198       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

3199     }
3200     \msg_error:nn{stex}{error/nomodule}
3201 }
3202
3203 \str_set:Nx \l_stex_module_name_str {
3204     \prop_item:Nn \l_stex_current_module_prop
3205     { name } / #2 - feature
3206 }
3207
3208 \str_set:Nx \l_stex_module_ns_str {
3209     \prop_item:Nn \l_stex_current_module_prop
3210     { ns }
3211 }
3212
3213
3214 \str_clear:N \l_tmpa_str
3215 \seq_clear:N \l_tmpa_seq
3216 \tl_clear:N \l_tmpa_tl
3217 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3218     origname = #2,
3219     name      = \l_stex_module_name_str ,
3220     ns        = \l_stex_module_ns_str ,
3221     imports   = \exp_not:o { \l_tmpa_seq } ,
3222     constants = \exp_not:o { \l_tmpa_seq } ,
3223     content   = \exp_not:o { \l_tmpa_tl } ,
3224     file      = \exp_not:o { \g_stex_currentfile_seq } ,
3225     lang      = \l_stex_module_lang_str ,
3226     sig       = \l_tmpa_str ,
3227     meta      = \l_tmpa_str ,
3228     feature   = #1 ,
3229 }
3230
3231 \stex_if_smsmode:TF {
3232     \stex_smsmode_set_codes:
3233 } {
3234     \begin{stex_annotate_env}{ feature:#1 }{}
3235     \stex_annotate_invisible:nnn{header}{}{ #3 }
3236 }
3237 }{
3238     \str_set:Nx \l_tmpa_str {
3239         c_stex_feature_
3240         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3241         \prop_item:Nn \l_stex_current_module_prop { name }
3242         _prop
3243     }
3244     \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3245     \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3246     \stex_if_smsmode:TF {
3247         \exp_args:Nx \stex_add_to_sms:n {
3248             \prop_gset_from_keyval:cn {
3249                 c_stex_feature_
3250                 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3251                 \prop_item:Nn \l_stex_current_module_prop { name }
3252                 _prop

```



```

3253     } {
3254         origname = #2,
3255         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3256         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3257         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3258         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3259         content   = \prop_item:cn { \l_tmpa_str } { content } ,
3260         file      = \prop_item:cn { \l_tmpa_str } { file } ,
3261         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
3262         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
3263         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
3264         feature   = \prop_item:cn { \l_tmpa_str } { feature }
3265     }
3266 }
3267 } {
3268     \end{stex_annotate_env}
3269 }
3270 }
3271

```

32.3 Features

structure

```

3272
3273 \prop_new:N \l_stex_all_structures_prop
3274
3275 \keys_define:nn { stex / features / structure } {
3276     name .str_set_x:N = \l__stex_features_structure_name_str ,
3277 }
3278
3279 \cs_new_protected:Nn \__stex_features_structure_args:n {
3280     \str_clear:N \l__stex_features_structure_name_str
3281     \keys_set:nn { stex / features / structure } { #1 }
3282 }
3283
3284 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3285 % \__stex_features_structure_args:n { ##1 }
3286 % \str_if_empty:NT \l__stex_features_structure_name_str {
3287 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3288 % }
3289 %} {
3290 %
3291 %}
3292
3293 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3294     \__stex_features_structure_args:n { #1 }
3295     \str_if_empty:NT \l__stex_features_structure_name_str {
3296         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3297     }
3298     \exp_args:Nnnx
3299     \begin{structural@feature}{ structure }
3300         { \l__stex_features_structure_name_str }{}
3301     \seq_clear:N \l_tmpa_seq

```

```

3302 \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3303
3304 }{
3305 \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3306 \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3307 \str_set:Nx \l_tmpa_str {
3308 \prop_item:Nn \l_stex_current_module_prop { ns } ?
3309 \prop_item:Nn \l_stex_current_module_prop { name }
3310 }
3311 \seq_map_inline:Nn \l_tmpa_seq {
3312 \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3313 }
3314 \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3315 \exp_args:Nnx
3316 \AddToHookNext { env / mathstructure / after }{
3317 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3318 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3319 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3320 \STEXexport {
3321 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3322 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3323 {\l_tmpa_str}
3324 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3325 {#2}{\l_tmpa_str}
3326 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3327 % \prop_item:Nn \l_stex_current_module_prop { origname },
3328 % \l_tmpa_str
3329 % }
3330 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3331 % #2,\l_tmpa_str
3332 % }
3333 % \tl_set:cx { #2 } {
3334 % \stex_invoke_structure:n { \l_tmpa_str }
3335 % }
3336 }
3337
3338 \end{structural@feature}
3339 % \g_stex_last_feature_prop
3340 }

```

\instantiate

```

3341 \seq_new:N \l__stex_features_structure_field_seq
3342 \str_new:N \l__stex_features_structure_field_str
3343 \str_new:N \l__stex_features_structure_def_tl
3344 \prop_new:N \l__stex_features_structure_prop
3345 \NewDocumentCommand \instantiate { m O{} m }{
3346 \stex_smsmode_set_codes:
3347 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3348 \prop_set_eq:Nc \l__stex_features_structure_prop {
3349 c_stex_feature_\l_tmpa_str _prop
3350 }
3351 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3352 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3353 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }

```

```

3354 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3355   \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3356   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3357     {} \l_tmpa_tl
3358   \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3359     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3360     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3361     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3362   }{
3363     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3364     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3365     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3366       \l_tmpa_tl
3367     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3368       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3369       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3370     }{
3371       \tl_clear:N \l_tmpb_tl
3372     }
3373   }
3374 }{
3375   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3376   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3377     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3378     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3379     \tl_clear:N \l_tmpa_tl
3380   }{
3381     % TODO throw error
3382   }
3383 }
3384 % \l_tmpa_str: name
3385 % \l_tmpa_tl: definiens
3386 % \l_tmpb_tl: notation
3387 \tl_if_empty:NT \l__stex_features_structure_field_str {
3388   % TODO throw error
3389 }
3390 \str_clear:N \l_tmpb_str
3391
3392 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3393 \seq_map_inline:Nn \l_tmpa_seq {
3394   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3395   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3396   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3397     \seq_map_break:n {
3398       \str_set:Nn \l_tmpb_str { ####1 }
3399     }
3400   }
3401 }
3402 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3403   \l_tmpb_str
3404
3405 \tl_if_empty:NTF \l_tmpb_tl {
3406   \tl_if_empty:NF \l_tmpa_tl {
3407     \exp_args:Nx \use:n {

```

```

3408         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3409     }
3410 }
3411 }{
3412     \tl_if_empty:NTF \l_tmpa_tl {
3413         \exp_args:Nx \use:n {
3414             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3415         }
3416     }
3417 }{
3418     \exp_args:Nx \use:n {
3419         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3420         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3421     }
3422 }
3423 }
3424 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3425 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3426 % #3/\l__stex_features_structure_field_str
3427 % \par
3428 % \expandafter\present\csname
3429 %     l_stex_symdecl_
3430 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
3431 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3432 % #3/\l__stex_features_structure_field_str
3433 % _prop
3434 % \endcsname
3435 }
3436
3437 \tl_clear:N \l__stex_features_structure_def_tl
3438
3439 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3440 \seq_map_inline:Nn \l_tmpa_seq {
3441     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3442     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3443     \exp_args:Nx \use:n {
3444         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3445
3446     }
3447 }
3448
3449 \prop_if_exist:cF {
3450     l_stex_symdecl_
3451     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3452     \prop_item:Nn \l_stex_current_module_prop {name} ?
3453     #3/\l_tmpa_str
3454     _prop
3455 }{
3456     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3457     \l_tmpb_str
3458     \exp_args:Nx \use:n {
3459         \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3460     }
3461 }

```

```

3462 }
3463
3464 \symdecl*[type={\STEXsymbol{module-type}}{
3465   \stex_term_math_oms:nnnn {
3466     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3467     \prop_item:Nn \l__stex_features_structure_prop {name}
3468     }{}{0}{}
3469   }{}{#3}
3470
3471   % TODO: -> sms file
3472
3473   \tl_set:cx{ #3 }{
3474     \stex_invoke_structure:nnn {
3475       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3476       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3477     } {
3478       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3479       \prop_item:Nn \l__stex_features_structure_prop {name}
3480     }
3481   }
3482
3483 }

```

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

\stex_invoke_structure:nnn

```

3484 % #1: URI of the instance
3485 % #2: URI of the instantiated module
3486 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3487   \tl_if_empty:nTF{ #3 }{
3488     \prop_set_eq:Nc \l__stex_features_structure_prop {
3489       c_stex_feature_ #2 _prop
3490     }
3491     \tl_clear:N \l_tmpa_tl
3492     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3493     \seq_map_inline:Nn \l_tmpa_seq {
3494       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3495       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3496       \cs_if_exist:cT {
3497         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3498       }{
3499         \tl_if_empty:NF \l_tmpa_tl {
3500           \tl_put_right:Nn \l_tmpa_tl {,}
3501         }
3502         \tl_put_right:Nx \l_tmpa_tl {
3503           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3504         }
3505       }
3506     }
3507     \exp_args:No \mathstruct \l_tmpa_tl
3508   }{
3509     \stex_invoke_symbol:n{#1/#3}
3510   }
3511 }

```

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

3512 `\endpackage`

Chapter 33

STEX -Statements Implementation

```
3513 <*package>
3514
3515 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3516
3517 \protected\def\ignorespacesandpars{
3518   \begingroup\catcode13=10\relax
3519   \@ifnextchar\par{
3520     \endgroup\expandafter\ignorespacesandpars\@gobble
3521   }{
3522     \endgroup
3523   }
3524 }
3525
3526 <@@=stex_statements>
3527
3528 Warnings and error messages
```

\titleemph

```
3528 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3529 \keys_define:nn {stex / definiendum }{
3530   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3531   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3532   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3533 }
3534 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3535   \str_clear:N \l__stex_statements_definiendum_root_str
3536   \tl_clear:N \l__stex_statements_definiendum_post_tl
3537   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3538 \keys_set:nn { stex / definiendum } { #1 }
3539 }
3540 \NewDocumentCommand \definiendum { 0{} m m } {
3541   \__stex_statements_definiendum_args:n { #1 }
3542   \stex_get_symbol:n { #2 }
3543   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3544   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3545     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3546       \tl_set:Nn \l_tmpa_tl { #3 }
3547     } {
3548       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3549       \tl_set:Nn \l_tmpa_tl {
3550         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3551       }
3552     }
3553   } {
3554     \tl_set:Nn \l_tmpa_tl { #3 }
3555   }
3556
3557   % TODO root
3558   \rustex_if:TF {
3559     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3560   } {
3561     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3562   }
3563 }
3564 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3565 \NewDocumentCommand \definame { 0{} m } {
3566   \__stex_statements_definiendum_args:n { #1 }
3567   % TODO: root
3568   \stex_get_symbol:n { #2 }
3569   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3570   \str_set:Nx \l_tmpa_str {
3571     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3572   }
3573   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3574   \rustex_if:TF {
3575     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3576       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3577     }
3578   } {
3579     \defemph@uri {
3580       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3581     } { \l_stex_get_symbol_uri_str }
3582   }
3583 }
3584 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3585
3586 \keys_define:nn {stex / sdefinition }{
3587   type      .str_set_x:N = \sdefinitiontype,
3588   id        .str_set_x:N = \sdefinitionid,
3589   title     .tl_set:N    = \sdefinitiontitle
3590 }
3591 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3592   \str_clear:N \sdefinitiontype
3593   \str_clear:N \sdefinitionid
3594   \tl_clear:N \sdefinitiontitle
3595   \keys_set:nn { stex / sdefinition }{ #1 }
3596 }
3597
3598 \NewDocumentEnvironment{sdefinition}{0{}}{
3599   \__stex_statements_sdefinition_args:n{ #1 }
3600   \stex_reactivate_macro:N \definiendum
3601   \stex_reactivate_macro:N \definame
3602   \stex_smsmode_set_codes:
3603   \clist_set:No \l_tmpa_clist \sdefinitiontype
3604   \tl_clear:N \l_tmpa_tl
3605   \clist_map_inline:Nn \l_tmpa_clist {
3606     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3607       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3608     }
3609   }
3610   \tl_if_empty:NTF \l_tmpa_tl {
3611     \__stex_statements_sdefinition_start:
3612   }{
3613     \l_tmpa_tl
3614   }
3615   \stex_ref_new_doc_target:n \sdefinitionid
3616   \stex_if_smsmode:F {
3617     \exp_args:Nnnx
3618     \begin{stex_annotate_env}{definition}{}
3619     \str_if_empty:NF \sdefinitiontype {
3620       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3621     }
3622   }
3623   }{
3624     \stex_if_smsmode:F {
3625       \end{stex_annotate_env}
3626     }
3627     \clist_set:No \l_tmpa_clist \sdefinitiontype
3628     \tl_clear:N \l_tmpa_tl
3629     \clist_map_inline:Nn \l_tmpa_clist {
3630       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3631         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3632       }
3633     }
3634     \tl_if_empty:NTF \l_tmpa_tl {
3635       \__stex_statements_sdefinition_end:
3636     }{
3637       \l_tmpa_tl

```

```

3638 }
3639 }

```

`\stexpatchdefinition`

```

3640 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3641   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3642     ~(\sdefinitiontitle)
3643   }~}
3644 }
3645 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3646
3647 \newcommand\stexpatchdefinition[3] [] {
3648   \str_set:Nx \l_tmpa_str{ #1 }
3649   \str_if_empty:NTF \l_tmpa_str {
3650     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3651     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3652   }{
3653     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3654     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3655   }
3656 }

```

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

`\inlinedef inline:`

```

3657 \NewDocumentCommand \inlinedef { m } {
3658   \begingroup
3659   \stex_reactivate_macro:N \definiendum
3660   \stex_reactivate_macro:N \definame
3661   \stex_ref_new_doc_target:n{
3662     #1
3663   }
3664 }

```

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

33.2 Assertions

`sassertion`

```

3665
3666 \keys_define:nn {stex / sassertion }{
3667   type      .str_set_x:N = \sassertiontype,
3668   id        .str_set_x:N = \sassertionid,
3669   title     .tl_set:N     = \sassertiontitle ,
3670   name      .str_set_x:N = \sassertionname
3671 }
3672 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3673   \str_clear:N \sassertiontype
3674   \str_clear:N \sassertionid
3675   \str_clear:N \sassertionname
3676   \tl_clear:N \sassertiontitle
3677   \keys_set:nn { stex / sassertion }{ #1 }
3678 }

```

```

3679
3680 \tl_new:N \g__stex_statements_aftergroup_tl
3681
3682 \NewDocumentEnvironment{sassertion}{0{}}{
3683   \__stex_statements_sassertion_args:n{ #1 }
3684   \stex_smsmode_set_codes:
3685   \clist_set:No \l_tmpa_clist \sassertiontype
3686   \tl_clear:N \l_tmpa_tl
3687   \clist_map_inline:Nn \l_tmpa_clist {
3688     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3689       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3690     }
3691   }
3692   \tl_if_empty:NTF \l_tmpa_tl {
3693     \__stex_statements_sassertion_start:
3694   }{
3695     \l_tmpa_tl
3696   }
3697   \stex_ref_new_doc_target:n \sassertionid
3698   \stex_if_smsmode:F {
3699     \exp_args:Nnnx
3700     \begin{stex_annotate_env}{assertion}{}
3701     \str_if_empty:NF \sassertiontype {
3702       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3703     }
3704   }
3705 }{
3706   \stex_if_smsmode:F {
3707     \end{stex_annotate_env}
3708   }
3709   \clist_set:No \l_tmpa_clist \sassertiontype
3710   \tl_clear:N \l_tmpa_tl
3711   \clist_map_inline:Nn \l_tmpa_clist {
3712     \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3713       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3714     }
3715   }
3716   \tl_if_empty:NTF \l_tmpa_tl {
3717     \__stex_statements_sassertion_end:
3718   }{
3719     \l_tmpa_tl
3720   }
3721   \str_if_empty:NF \sassertionname {
3722     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3723       \symdecl*{\sassertionname}
3724     }
3725     \aftergroup\g__stex_statements_aftergroup_tl
3726   }
3727 }

```

\stexpatchassertion

```

3728
3729 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3730   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3731     (\sassertiontitle)
3732   }~}
3733 }
3734 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3735
3736 \newcommand\stexpatchassertion[3] [] {
3737   \str_set:Nx \l_tmpa_str{ #1 }
3738   \str_if_empty:NTF \l_tmpa_str {
3739     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3740     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3741   }{
3742     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3743     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3744   }
3745 }

```

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

\inlineass inline:

```

3746 \NewDocumentCommand \inlineass { m } {
3747   \begingroup
3748   \stex_ref_new_doc_target:n{
3749     #1
3750   }
3751 }

```

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

33.3 Examples

sexample

```

3752
3753 \keys_define:nn {stex / sexample }{
3754   type      .str_set_x:N = \exampletype,
3755   id        .str_set_x:N = \sexampleid,
3756   title     .tl_set:N = \sexampletile,
3757   for       .clist_set:N = \sexamplefor,
3758 }
3759 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3760   \str_clear:N \sexampletype
3761   \str_clear:N \sexampleid
3762   \tl_clear:N \sexampletile
3763   \clist_clear:N \sexamplefor
3764   \keys_set:nn { stex / sexample }{ #1 }
3765 }
3766
3767 \NewDocumentEnvironment{sexample}{0{}}{
3768   \__stex_statements_sexample_args:n{ #1 }
3769   \stex_smsmode_set_codes:
3770   \clist_set:Nn \l_tmpa_clist \sexampletype
3771   \tl_clear:N \l_tmpa_tl
3772   \clist_map_inline:Nn \l_tmpa_clist {
3773     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3774     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3775   }
3776 }
3777 \tl_if_empty:NTF \l_tmpa_tl {
3778   \__stex_statements_sexample_start:
3779 }{
3780   \l_tmpa_tl
3781 }
3782 \stex_ref_new_doc_target:n \sexampleid
3783 \stex_if_smsmode:F {
3784   \seq_clear:N \l_tmpa_seq
3785   \clist_map_inline:Nn \sexamplefor {
3786     \str_if_eq:nnF{ ##1 }{{}{
3787       \stex_get_symbol:n { ##1 }
3788       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3789         \l_stex_get_symbol_uri_str
3790       }
3791     }
3792   }
3793   \exp_args:Nnnx
3794   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3795   \str_if_empty:NF \sexamplotype {
3796     \stex_annotate_invisible:nnn{type}{\sexampletype}{}
3797   }
3798 }
3799 }{
3800   \stex_if_smsmode:F {
3801     \end{stex_annotate_env}
3802   }
3803   \clist_set:Nn \l_tmpa_clist \sexamplotype
3804   \tl_clear:N \l_tmpa_tl
3805   \clist_map_inline:Nn \l_tmpa_clist {
3806     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3807       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3808     }
3809   }
3810   \tl_if_empty:NTF \l_tmpa_tl {
3811     \__stex_statements_sexample_end:
3812   }{
3813     \l_tmpa_tl
3814   }
3815 }

```

\stexpatchexample

```

3816
3817 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3818   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3819     (\sexamplotype)
3820   }~}
3821 }
3822 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3823
3824 \newcommand\stexpatchexample[3]{} {
3825   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3826 \str_if_empty:NTF \l_tmpa_str {
3827   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3828   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3829 }{
3830   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3831   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3832 }
3833 }

```

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

`\inlineex` inline:

```

3834 \NewDocumentCommand \inlineex { m } {
3835   \begingroup
3836   \stex_ref_new_doc_target:n{
3837     #1
3838   }
3839 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

3840 \keys_define:nn { stex / sparagraph } {
3841   id      .str_set_x:N = \sparagraphid ,
3842   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3843   type    .str_set_x:N = \sparagraphtype ,
3844   for     .str_set_x:N = \sparagraphfor ,
3845   from    .tl_set_x:N  = \sparagraphfrom ,
3846   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
3847   name    .str_set:N   = \sparagraphname
3848 }
3849
3850 \cs_new_protected:Nn \stex_sparagraph_args:n {
3851   \tl_clear:N \l_stex_sparagraph_title_tl
3852   \tl_clear:N \sparagraphfrom
3853   \tl_clear:N \l_stex_sparagraph_start_tl
3854   \str_clear:N \sparagraphid
3855   \str_clear:N \sparagraphtype
3856   \str_clear:N \sparagraphfor
3857   \str_clear:N \sparagraphname
3858   \keys_set:nn { stex / sparagraph }{ #1 }
3859 }
3860 \newif\if@in@omtext\@in@omtextfalse
3861
3862 \NewDocumentEnvironment {sparagraph} { 0{} } {
3863   \stex_sparagraph_args:n { #1 }
3864   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3865     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3866   }{
3867     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3868   }

```

```

3869 \@in@omtexttrue
3870 \stex_smsmode_set_codes:
3871 \clist_set:No \l_tmpa_clist \sparagraphtype
3872 \tl_clear:N \l_tmpa_tl
3873 \clist_map_inline:Nn \l_tmpa_clist {
3874   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3875     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3876   }
3877 }
3878 \tl_if_empty:NTF \l_tmpa_tl {
3879   \__stex_statements_sparagraph_start:
3880 }{
3881   \l_tmpa_tl
3882 }
3883 \stex_ref_new_doc_target:n \sparagraphid
3884 \stex_if_smsmode:F {
3885   \exp_args:Nnnx
3886   \begin{stex_annotate_env}{paragraph}{}
3887   \str_if_empty:NF \sparagraphtype {
3888     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
3889   }
3890 }
3891 \ignorespacesandpars
3892 }{
3893   \stex_if_smsmode:F {
3894     \end{stex_annotate_env}
3895   }
3896   \clist_set:No \l_tmpa_clist \sparagraphtype
3897   \tl_clear:N \l_tmpa_tl
3898   \clist_map_inline:Nn \l_tmpa_clist {
3899     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
3900       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
3901     }
3902   }
3903   \tl_if_empty:NTF \l_tmpa_tl {
3904     \__stex_statements_sparagraph_end:
3905   }{
3906     \l_tmpa_tl
3907   }
3908   \str_if_empty:NF \sparagraphname {
3909     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3910       \symdecl*{\sparagraphname}
3911     }
3912     \aftergroup\g__stex_statements_aftergroup_tl
3913   }
3914 }

```

\stexpatchparagraph

```

3915
3916 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
3917   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3918     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
3919       \titleemph{\l_stex_sparagraph_title_tl}:~
3920     }

```

```

3921   }{
3922     \titleemph{\l_stex_sparagraph_start_tl}~
3923   }
3924 }
3925 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
3926
3927 \newcommand\stexpatchparagraph[3] [] {
3928   \str_set:Nx \l_tmpa_str{ #1 }
3929   \str_if_empty:NTF \l_tmpa_str {
3930     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
3931     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
3932   }{
3933     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
3934     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
3935   }
3936 }

```

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

symboldoc

```

3937 \NewDocumentEnvironment{symboldoc}{ m }{
3938   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3939   \seq_clear:N \l_tmpb_seq
3940   \seq_map_inline:Nn \l_tmpa_seq {
3941     \str_if_eq:nnF{ ##1 }{}{
3942       \stex_get_symbol:n { ##1 }
3943       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3944         \l_stex_get_symbol_uri_str
3945       }
3946     }
3947   }
3948   \par
3949   \exp_args:Nnnx
3950   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3951 }{
3952   \end{stex_annotate_env}
3953 }
3954 \</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

```
3955 <*package>
3956 <@@=stex_sproof>
3957
3958 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3959
```

34.2 Proofs

We first define some keys for the proof environment.

```
3960 \keys_define:nn { stex / spf } {
3961   id          .str_set:N = \l__stex_sproof_spf_id_str,
3962   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3963   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3964   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3965   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3966   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3967   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3968   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3969   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3970   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3971 }
3972 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3973   \str_clear:N \l__stex_sproof_spf_id_str
3974   \tl_clear:N \l__stex_sproof_spf_display_tl
3975   \tl_clear:N \l__stex_sproof_spf_for_tl
3976   \tl_clear:N \l__stex_sproof_spf_from_tl
3977   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3978   \tl_clear:N \l__stex_sproof_spf_type_tl
3979   \tl_clear:N \l__stex_sproof_spf_title_tl

```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

3980 \tl_clear:N \l__stex_sproof_spf_continues_tl
3981 \tl_clear:N \l__stex_sproof_spf_functions_tl
3982 \tl_clear:N \l__stex_sproof_spf_method_tl
3983 \keys_set:nn { stex / spf }{ #1 }
3984 }

```

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

```

3985 \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.

```

3986 \newcount\count_ten
3987 \newenvironment{pst@with@label}[1]{
3988   \edef\pst@label{#1}
3989   \advance\count_ten by 1\relax
3990   \count_ten=1
3991 }{
3992   \advance\count_ten by -1\relax
3993 }

```

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

```

3994 \def\the@pst@label{
3995   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3996 }

```

(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.

```

3997 \keys_define:nn { stex / pstlabel }{
3998   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3999   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4000   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4001 }
4002 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4003 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4004 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4005 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4006 }
4007 \__stex_sproof_pstlabel_args:n {}
4008 \newcommand\setpstlabelstyle[1]{
4009   \__stex_sproof_pstlabel_args:n {#1}
4010 }
4011 \newcommand\setpstlabelstyledefault{%
4012   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4013 }

```

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

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

```

4014 \ExplSyntaxOff
4015 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4016 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4017 \def\pst@make@label@short#1#2{#2}
4018 \def\pst@make@label@empty#1#2{}
4019 \ExplSyntaxOn
4020 \def\pstlabelstyle#1{%
4021   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4022 }%
4023 \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.

```

4024 \def\next@pst@label{%
4025   \global\advance\count\count10 by 1%
4026 }%

```

(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

```

4027 \def\sproof@box{
4028   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4029 }
4030 \def\spf@proofend{\sproof@box}
4031 \def\sproofend{
4032   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4033     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4034   }
4035 }
4036 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4037 \def\spf@proofsketch@kw{Proof Sketch}
4038 \def\spf@proof@kw{Proof}
4039 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4040 \cs_if_exist:NT \bbl@loaded {
4041   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4042   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4043     \input{proof-ngerman.ldf}
4044   }
4045   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4046     \input{proof-finnish.ldf}
4047   }
4048   \clist_if_in:NnT \l_tmpa_clist {french}{
4049     \input{proof-french.ldf}
4050   }
4051   \clist_if_in:NnT \l_tmpa_clist {russian}{
4052     \input{proof-russian.ldf}
4053   }
4054 }
4055

```

spfsketch

```

4056 \newcommand\spfsketch[2][]{
4057   \__stex_sproof_spf_args:n{#1}
4058   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4059     \titleemph{
4060       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4061         \spf@proofsketch@kw
4062       }{
4063         \l__stex_sproof_spf_type_tl
4064       }
4065     }:
4066   }
4067   {~#2}
4068   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4069   \sproofend
4070 }

```

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

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

```

4071 \newenvironment{spfeq}[2][]{
4072   \__stex_sproof_spf_args:n{#1}
4073   %\sref@target
4074   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4075     \titleemph{
4076       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4077         \spf@proof@kw
4078       }{
4079         \l__stex_sproof_spf_type_tl
4080       }
4081     }:

```

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

¹⁵EDNOTE: document above

```

4082 }
4083 {~#2}
4084 \begin{displaymath}\begin{array}{rcll}
4085 }{
4086 \end{array}\end{displaymath}
4087 }

```

(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.

```

4088 \newenvironment{spf@proof}[2][]{
4089   \__stex_sproof_spf_args:n{#1}
4090   %\sref@target
4091   \count_ten=10
4092   \par\noindent
4093   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4094     \titleemph{
4095       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4096         \spf@proof@kw
4097       }{
4098         \l__stex_sproof_spf_type_tl
4099       }
4100     }:
4101   }
4102   {~#2}
4103   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
4104   \def\pst@label{}
4105   \newcount\pst@count% initialize the labeling mechanism
4106   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4107   }{
4108     \end{pst@with@label}\end{description}
4109   }
4110 \newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4111 \newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4112 \newcommand\spfidea[2][]{
4113   \__stex_sproof_spf_args:n{#1}
4114   \titleemph{
4115     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4116       \l__stex_sproof_spf_type_tl
4117     }:
4118   }~#2
4119   \sproofend
4120 }

```

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

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

spfstep 16

```

4121 \newenvironment{spfstep}[1][]{
4122   \_stex_sproof_spf_args:n{#1}
4123   \@in@omtexttrue
4124   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4125     \item[\the@pst@label]
4126   }
4127   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4128     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4129   }
4130   %\sref@label@id{\pst@label}
4131   \ignorespacesandpars
4132 }{
4133   \next@pst@label\ignorespacesandpars
4134 }

```

sproofcomment

```

4135 \newenvironment{sproofcomment}[1][]{
4136   \_stex_sproof_spf_args:n{#1}
4137   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4138     \item[\the@pst@label]
4139   }
4140 }{
4141   \next@pst@label
4142 }

```

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.

```

4143 \newenvironment{subproof}[2][]{
4144   \_stex_sproof_spf_args:n{#1}
4145   \def\@test{#2}
4146   \ifx\@test\empty\else
4147     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4148       \item[\the@pst@label]
4149     }{#2}
4150   \fi
4151   \begin{pst@with@label}{\pst@label,\number\count_ten}
4152 }{
4153   \end{pst@with@label}\next@pst@label
4154 }

```

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

```

4155 \newenvironment{spfcases}[2][]{
4156   \def\@test{#1}
4157   \ifx\@test\empty
4158     \begin{subproof}[method=by-cases]{#2}
4159   \else
4160     \begin{subproof}[#1,method=by-cases]{#2}
4161   \fi
4162 }{

```

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

```

4163 \end{subproof}
4164 }

```

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

```

4165 \newenvironment{spfcase}[2] [] {
4166   \__stex_sproof_spf_args:n{#1}
4167   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4168     \item[\the@pst@label]
4169   }
4170   \def\@test{#2}
4171   \ifx\@test\@empty
4172   \else
4173     {\titleemph{#2}:~}
4174   \fi
4175   \begin{pst@with@label}{\pst@label,\number\count_ten}
4176 }{
4177   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4178     \sproofend
4179   }
4180   \end{pst@with@label}
4181   \next@pst@label
4182 }

```

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

```

4183 \newcommand\spfcasesketch[3] [] {
4184   \__stex_sproof_spf_args:n{#1}
4185   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4186     \item[\the@pst@label]
4187   }
4188   \def\@test{#2}
4189   \ifx\@test\@empty
4190   \else
4191     {\titleemph{#2}:~}
4192   \fi#3
4193   \next@pst@label
4194 }%

```

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.

```

4195 \keys_define:nn { stex / just }{
4196   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4197   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
4198   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
4199   args        .tl_set:N   = \l__stex_sproof_just_args_tl
4200 }

```

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

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

justification

```
4201 \newenvironment{justification}[1] [] {}{}
```

\premise

```
4202 \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.

```
4203 \newcommand\justarg[2] [] {#2}
```

```
4204 \end{package}
```

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

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

Chapter 35

STEX -Others Implementation

```
4205 <*package>
4206
4207 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4208
4209 <@@=stex_others>
    Warnings and error messages
4210 % None

\MSC Math subject classifier

4211 \NewDocumentCommand \MSC {m} {
4212 % TODO
4213 }

(End definition for \MSC. This function is documented on page 20.)
    Patching tikzinput, if loaded
4214 \@ifpackageloaded{tikzinput}{
4215 \RequirePackage{stex-tikzinput}
4216 }{}
4217 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4218 \*package>
4219 \@@=stex_modules>
4220
4221 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4222
4223 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4224 \begingroup
4225 \stex_module_setup:nn{
4226   ns=\c_stex_metatheory_ns_str,
4227   meta=NONE
4228 }{Metatheory}
4229 \stex_reactivate_macro:N \symdecl
4230 \stex_reactivate_macro:N \notation
4231 \stex_reactivate_macro:N \symdef
4232 \ExplSyntaxOff
4233 \csname stex_suppress_html:n\endcsname{
4234   % is-a (a:A, a \in A, a is an A, etc.)
4235   \symdecl[args=ai]{isa}
4236   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4237   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4238   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4239
4240   % bind (\forall, \Pi, \lambda etc.)
4241   \symdecl[args=Bi]{bind}
4242   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4243   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4244   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4245
4246   % dummy variable
4247   \symdecl{dummyvar}
4248   \notation[underscore]{dummyvar}{\comp\_}
4249   \notation[dot]{dummyvar}{\comp\cdot}
4250   \notation[dash]{dummyvar}{\comp{\rm --}}
4251
4252   %fromto (function space, Hom-set, implication etc.)
```

```

4253 \symdecl[args=ai]{fromto}
4254 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4255 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4256
4257 % mapto (lambda etc.)
4258 %\symdecl[args=Bi]{mapto}
4259 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4260 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4261 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4262
4263 % function/operator application
4264 \symdecl[args=ia]{apply}
4265 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4266 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4267
4268 % ‘type’ of all collections (sets, classes, types, kinds)
4269 \symdecl{collection}
4270 \notation[U]{collection}{\comp{\mathcal{U}}}
4271 \notation[set]{collection}{\comp{\textsf{Set}}}
4272
4273 % sequences
4274 \symdecl[args=1]{seqtype}
4275 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4276
4277 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4278 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4279
4280 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4281 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4282 % ^ superceded by \aseqfromto and \livar/\uivar
4283
4284 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4285 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4286 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}}{#1\comp,#2}
4287
4288 % letin (‘let’, local definitions, variable substitution)
4289 \symdecl[args=bii]{letin}
4290 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}\;#2\; \comp{\rm in}}{#3}
4291 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4292 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4293
4294 % structures
4295 \symdecl*[args=1]{module-type}
4296 \notation{module-type}{\mathtt{MOD} #1}
4297 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4298 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4299
4300 }
4301 \ExplSyntaxOn
4302 \stex_add_to_current_module:n{
4303   \let\nappa\apply
4304   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4305   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4306   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4307     \def\uivar{\csname sequence-index\endcsname[ui]}
4308     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4309     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4310     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4311   }
4312   \__stex_modules_end_module:
4313   \endgroup
4314 \endpackage

```

Chapter 37

Tikzinput Implementation

```
4315 <*package>
4316
4317 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4318
4319 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4320 \RequirePackage{l3keys2e}
4321
4322 \keys_define:nn { tikzinput } {
4323   image .bool_set:N = \c_tikzinput_image_bool,
4324   image .default:n = false ,
4325   unknown .code:n = {}
4326 }
4327
4328 \ProcessKeysOptions { tikzinput }
4329
4330 \bool_if:NTF \c_tikzinput_image_bool {
4331   \RequirePackage{graphicx}
4332
4333   \providecommand\usetikzlibrary[]{}
4334   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4335 }{
4336   \RequirePackage{tikz}
4337   \RequirePackage{standalone}
4338
4339   \newcommand \tikzinput [2] [] {
4340     \setkeys{Gin}{#1}
4341     \ifx \Gin@ewidth \Gin@exclamation
4342       \ifx \Gin@eheight \Gin@exclamation
4343         \input { #2 }
4344       \else
4345         \resizebox{!}{ \Gin@eheight }{
4346           \input { #2 }
4347         }
4348       \fi
4349     \else
4350       \ifx \Gin@eheight \Gin@exclamation
4351         \resizebox{ \Gin@ewidth }{!}{
4352           \input { #2 }
```

```

4353     }
4354     \else
4355         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4356             \input { #2 }
4357         }
4358     \fi
4359 \fi
4360 }
4361 }
4362
4363 \newcommand \ctikzinput [2] [] {
4364     \begin{center}
4365         \tikzinput [1] {#2}
4366     \end{center}
4367 }
4368
4369 \@ifpackageloaded{stex}{
4370     \RequirePackage{stex-tikzinput}
4371 }{}
4372
4373 </package>
4374 <*stex>
4375 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4376 \RequirePackage{stex}
4377 \RequirePackage{tikzinput}
4378
4379 \newcommand\mhtikzinput [2] [] {%
4380     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4381     \stex_in_repository:nn\Gin@mhrepos{
4382         \tikzinput [1]{\mhpath{##1}{#2}}
4383     }
4384 }
4385 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4386 </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.

```
4387 \*cls)
4388 \@@=document_structure)
4389 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4390 \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`

```
4391 \keys_define:nn{ document-structure / pkg }{
4392   class      .str_set_x:N = \c_document_structure_class_str,
4393   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4394   report     .code:n      = {
4395     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4396     \str_set:Nn \c_document_structure_class_str {report}
4397   },
4398   book       .code:n      = {
4399     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4400     \str_set:Nn \c_document_structure_class_str {book}
4401   },
4402   bookpart   .code:n      = {
4403     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4404     \str_set:Nn \c_document_structure_class_str {book}
4405     \str_set:Nn \c_document_structure_topsect_str {chapter}
4406   },
```

```

4407 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4408 unknown     .code:n      = {
4409   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4410 }
4411 }
4412 \ProcessKeysOptions{ document-structure / pkg }
4413 \str_if_empty:NT \c_document_structure_class_str {
4414   \str_set:Nn \c_document_structure_class_str {article}
4415 }
4416 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4417   {\c_document_structure_class_str}
4418

```

38.3 Beefing up the document environment

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

```

4419 \RequirePackage{omdoc}
4420 \bool_if:NF \c_document_structure_minimal_bool {
4421   \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

```

4422 \keys_define:nn { document-structure / document }{
4423   id .str_set_x:N = \c_document_structure_document_id_str
4424 }
4425 \let\__document_structure_orig_document=\document
4426 \renewcommand{\document}[1][]{
4427   \keys_set:nn{ document-structure / document }{ #1 }
4428   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4429   \__document_structure_orig_document
4430 }

```

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

```

4431 }
4432 \</cls>

```

38.4 Implementation: OMDoc Package

```

4433 \*package>
4434 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4435 \RequirePackage{expl-keys-compact,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


```

4436
4437 \keys_define:nn{ document-structure / pkg }{
4438   class      .str_set_x:N = \c_document_structure_class_str,
4439   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4440   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4441 }
4442 \ProcessKeysOptions{ document-structure / pkg }
4443 \str_if_empty:NT \c_document_structure_class_str {
4444   \str_set:Nn \c_document_structure_class_str {article}
4445 }
4446 \str_if_empty:NT \c_document_structure_topsect_str {
4447   \str_set:Nn \c_document_structure_topsect_str {section}
4448 }

```

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

```

4449 \RequirePackage{xspace}
4450 \RequirePackage{comment}
4451 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4452 \@ifpackageloaded{babel}{
4453   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4454   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4455     \input{omdoc-ngerman.ldf}
4456   }
4457 }{}
4458 %\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.

```

4459 \int_new:N \l_document_structure_section_level_int
4460 \str_case:VnF \c_document_structure_topsect_str {
4461   {part}{
4462     \int_set:Nn \l_document_structure_section_level_int {0}
4463   }
4464   {chapter}{
4465     \int_set:Nn \l_document_structure_section_level_int {1}
4466   }
4467 }{
4468   \str_case:VnF \c_document_structure_class_str {
4469     {book}{
4470       \int_set:Nn \l_document_structure_section_level_int {0}
4471     }
4472     {report}{
4473       \int_set:Nn \l_document_structure_section_level_int {0}
4474     }
4475   }{
4476     \int_set:Nn \l_document_structure_section_level_int {2}
4477   }
4478 }

```

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

```
4479 \def\current@section@level{document}%
4480 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4481 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4482 \cs_new_protected:Npn \skipomgroup {
4483   \ifcase\l_document_structure_section_level_int
4484   \or\stepcounter{part}
4485   \or\stepcounter{chapter}
4486   \or\stepcounter{section}
4487   \or\stepcounter{subsection}
4488   \or\stepcounter{subsubsection}
4489   \or\stepcounter{paragraph}
4490   \or\stepcounter{subparagraph}
4491   \fi
4492 }
```

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

`blindomgroup`

```
4493 \newcommand\at@begin@blindomgroup[1]{%
4494 \newenvironment{blindomgroup}
4495 {
4496   \int_incr:N\l_document_structure_section_level_int
4497   \at@begin@blindomgroup\l_document_structure_section_level_int
4498 }{}}
```

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

```
4499 \newcommand\omgroup@nonum[2]{
4500   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4501   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4502 }
```

(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.

```
4503 \newcommand\omgroup@num[2]{
```

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

```

4504 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4505   \@nameuse{#1}{#2}
4506 }{
4507   \cs_if_exist:NTF\rdfmata@sectioning{
4508     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4509   }{
4510     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4511   }
4512 }
4513 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4514 }

```

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

omgroup

```

4515 \keys_define:nn { document-structure / omgroupp }{
4516   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4517   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4518   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4519   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4520   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4521   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4522   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4523   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4524   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4525   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4526 }
4527 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4528   \str_clear:N \l__document_structure_omgroup_id_str
4529   \str_clear:N \l__document_structure_omgroup_date_str
4530   \clist_clear:N \l__document_structure_omgroup_creators_clist
4531   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4532   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4533   \tl_clear:N \l__document_structure_omgroup_type_tl
4534   \tl_clear:N \l__document_structure_omgroup_short_tl
4535   \tl_clear:N \l__document_structure_omgroup_display_tl
4536   \tl_clear:N \l__document_structure_omgroup_intro_tl
4537   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4538   \keys_set:nn { document-structure / omgroupp } { #1 }
4539 }

```

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.

```

4540 \newif\if@mainmatter\@mainmattertrue
4541 \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.

```

4542 \keys_define:nn { document-structure / sectioning }{
4543   name .str_set_x:N = \l__document_structure_sect_name_str ,
4544   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4545   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4546   num .bool_set:N = \l__document_structure_sect_num_bool ,
4547 }

```

```

4548 \cs_new_protected:Nn \l__document_structure_sect_args:n {
4549   \str_clear:N \l__document_structure_sect_name_str
4550   \str_clear:N \l__document_structure_sect_ref_str
4551   \bool_set_false:N \l__document_structure_sect_clear_bool
4552   \bool_set_false:N \l__document_structure_sect_num_bool
4553   \keys_set:nn { document-structure / sectioning } { #1 }
4554 }
4555 \newcommand\omdoc@sectioning[3][]{
4556   \l__document_structure_sect_args:n {#1}
4557   \let\omdoc@sect@name\l__document_structure_sect_name_str
4558   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4559   \if@mainmatter% numbering not overridden by frontmatter, etc.
4560     \bool_if:NTF \l__document_structure_sect_num_bool {
4561       \omgroup@num{#2}{#3}
4562     }{
4563       \omgroup@nonum{#2}{#3}
4564     }
4565     \def\current@section@level{\omdoc@sect@name}
4566   \else
4567     \omgroup@nonum{#2}{#3}
4568   \fi
4569 }% 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.

```

4570 \newcommand\omgroup@redefine@addtocontents[1]{%
4571   %\edef\__document_structureimport{#1}%
4572   %\@for\@I:=\__document_structureimport\do{%
4573     %\edef\@path{\csname module@\@I @path\endcsname}%
4574     %\@ifundefined{tf@toc}\relax%
4575     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4576   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4577   %\def\addcontentsline##1##2##3{%
4578     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4579   %\else% hyperref.sty not loaded
4580   %\def\addcontentsline##1##2##3{%
4581     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4582   %\fi
4583 }% 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.

```

4584 \int_new:N \l__document_structure_omgroup_level_int
4585 \newenvironment{omgroup}[2][]{% keys, title
4586 {
4587   \l__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

4588 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4589   \omgroup@redefine@addtocontents{
4590     %\@ifundefined{module@id}\used@modules%
4591     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4592     }
4593 }

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

4594 \int_incr:N \l_document_structure_omgroup_level_int
4595 \int_incr:N \l_document_structure_section_level_int
4596 \ifcase\l_document_structure_section_level_int
4597   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4598   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4599   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4600   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4601   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4602   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4603   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4604 \fi
4605 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4606 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4607 }% for customization
4608 {}

```

and finally, we localize the sections

```

4609 \newcommand\omdoc@part@kw{Part}
4610 \newcommand\omdoc@chapter@kw{Chapter}
4611 \newcommand\omdoc@section@kw{Section}
4612 \newcommand\omdoc@subsection@kw{Subsection}
4613 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4614 \newcommand\omdoc@paragraph@kw{paragraph}
4615 \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`

```

4616 \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 environ-
ments, we save their behavior (possibly defining it) in orig@*matter macros and make
them undefined (so that we can define the environments).

4617 \cs_if_exist:NTF\frontmatter{
4618   \let\__document_structure_orig_frontmatter\frontmatter
4619   \let\frontmatter\relax
4620 }{
4621   \tl_set:Nn\__document_structure_orig_frontmatter{
4622     \clearpage
4623     \@mainmatterfalse
4624     \pagenumbering{roman}
4625   }
4626 }
4627 \cs_if_exist:NTF\backmatter{

```

```

4628 \let\__document_structure_orig_backmatter\backmatter
4629 \let\backmatter\relax
4630 }{
4631 \tl_set:Nn\__document_structure_orig_backmatter{
4632 \clearpage
4633 \@mainmatterfalse
4634 \pagenumbering{roman}
4635 }
4636 }

```

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.

```

4637 \newenvironment{frontmatter}{
4638 \__document_structure_orig_frontmatter
4639 }{
4640 \cs_if_exist:NTF\mainmatter{
4641 \mainmatter
4642 }{
4643 \clearpage
4644 \@mainmattertrue
4645 \pagenumbering{arabic}
4646 }
4647 }

```

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

```

4648 \newenvironment{backmatter}{
4649 \__document_structure_orig_backmatter
4650 }{
4651 \cs_if_exist:NTF\mainmatter{
4652 \mainmatter
4653 }{
4654 \clearpage
4655 \@mainmattertrue
4656 \pagenumbering{arabic}
4657 }
4658 }

```

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

```

4659 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4660 \def \c__document_structure_document_str{document}
4661 \newcommand\afterprematurestop{}
4662 \def\prematurestop@endomgroup{
4663 \unless\ifx\@currenvir\c__document_structure_document_str
4664 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4665 \expandafter\prematurestop@endomgroup
4666 \fi
4667 }
4668 \providecommand\prematurestop{

```

```

4669 \message{Stopping~sTeX~processing~prematurely}
4670 \prematurestop@endomgroup
4671 \afterprematurestop
4672 \end{document}
4673 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

4674 \RequirePackage{etoolbox}
4675 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4676 \newrobustcmd\useSGvar[1]{%
4677   \@ifundefined{sTeX@Gvar@#1}
4678   {\PackageError{omdoc}
4679     {The sTeX Global variable #1 is undefined}
4680     {set it with \protect\setSGvar}}
4681   \@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.

```

4682 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4683   \@ifundefined{sTeX@Gvar@#1}
4684   {\PackageError{omdoc}
4685     {The sTeX Global variable #1 is undefined}
4686     {set it with \protect\setSGvar}}
4687   {\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.

```
4688 \*cls)
4689 \@@=mikoslides)
4690 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4691 \RequirePackage{l3keys2e,expl-keystr-compatible}
4692
4693 \keys_define:nn{mikoslides / cls}{
4694   class .code:n = {
4695     \PassOptionsToClass{\CurrentOption}{omdoc}
4696     \str_if_eq:nnT{#1}{book}{
4697       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4698     }
4699     \str_if_eq:nnT{#1}{report}{
4700       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4701     }
4702   },
4703   notes .bool_set:N = \c__mikoslides_notes_bool ,
4704   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4705   unknown .code:n = {
4706     \PassOptionsToClass{\CurrentOption}{omdoc}
4707     \PassOptionsToClass{\CurrentOption}{beamer}
4708     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4709   }
4710 }
4711 \ProcessKeysOptions{ mikoslides / cls }
4712 \bool_if:NTF \c__mikoslides_notes_bool {
4713   \PassOptionsToPackage{notes=true}{mikoslides}
4714 }{
4715   \PassOptionsToPackage{notes=false}{mikoslides}
4716 }
4717 \</cls)
```


now we do the same for the mikoslides package.

```

4718 <*package>
4719 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4720 \RequirePackage{l3keys2e,expl-keystr-compat}
4721
4722 \keys_define:nn{mikoslides / pkg}{
4723   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4724   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4725   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4726   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4727   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4728   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4729   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4730   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4731   unknown      .code:n      = {
4732     \PassOptionsToClass{\CurrentOption}{stex}
4733     \PassOptionsToClass{\CurrentOption}{tikzinput}
4734   }
4735 }
4736 \ProcessKeysOptions{ mikoslides / pkg }
4737 \newif\ifnotes
4738 \bool_if:NTF \c__mikoslides_notes_bool {
4739   \notesttrue
4740 }{
4741   \notesfalse
4742 }
4743

```

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

```

4744 \str_if_empty:NTF \c__mikoslides_topsect_str {
4745   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4746 }{
4747   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4748 }
4749 </package>

```

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

```

4750 <*cls>
4751 \bool_if:NTF \c__mikoslides_notes_bool {
4752   \LoadClass{omdoc}
4753 }{
4754   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4755   \newcounter{Item}
4756   \newcounter{paragraph}
4757   \newcounter{subparagraph}
4758   \newcounter{Hfootnote}
4759   \RequirePackage{omdoc}
4760 }

```

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

```

4761 \RequirePackage{mikoslides}
4762 </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.

```

4763 \*package>
4764 \bool_if:NT \c__mikoslides_notes_bool {
4765   \RequirePackage{a4wide}
4766   \RequirePackage{marginnote}
4767   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4768   \RequirePackage{mdframed}
4769   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4770   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4771 }
4772 \RequirePackage{stex-compatibility}
4773 \RequirePackage{stex-tikzinput}
4774 \RequirePackage{etoolbox}
4775 \RequirePackage{amssymb}
4776 \RequirePackage{amsmath}
4777 \RequirePackage{comment}
4778 \RequirePackage{textcomp}
4779 \RequirePackage{url}
4780 \RequirePackage{graphicx}
4781 \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

```

4782 \bool_if:NT \c__mikoslides_notes_bool {
4783   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
4784 }

```

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

```

4785 \newcounter{slide}
4786 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4787 \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.

```

4788 \bool_if:NTF \c__mikoslides_notes_bool {
4789   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
4790 }{
4791   \excludecomment{note}
4792 }

```

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.

```
4793 \bool_if:NT \c__mikoslides_notes_bool {
4794   \newlength{\slideframewidth}
4795   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4796 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4797   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4798     \bool_set_true:N #1
4799   }{
4800     \bool_set_false:N #1
4801   }
4802 }
4803 \keys_define:nn{mikoslides / frame}{
4804   label .str_set_x:N = \l__mikoslides_frame_label_str,
4805   allowframebreaks .code:n = {
4806     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4807   },
4808   allowdisplaybreaks .code:n = {
4809     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4810   },
4811   fragile .code:n = {
4812     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4813   },
4814   shrink .code:n = {
4815     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4816   },
4817   squeeze .code:n = {
4818     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4819   },
4820   t .code:n = {
4821     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4822   },
4823 }
4824 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4825   \str_clear:N \l__mikoslides_frame_label_str
4826   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4827   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4828   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4829   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4830   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4831   \bool_set_true:N \l__mikoslides_frame_t_bool
4832   \keys_set:nn { mikoslides / frame }{ #1 }
4833 }
```

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

```
4834 \renewenvironment{frame}[1][]{
4835   \__mikoslides_frame_args:n{#1}
4836   \sffamily
4837   \stepcounter{slide}
4838   \def\@currentlabel{\theslide}
4839   \str_if_empty:NF \l__mikoslides_frame_label_str {
4840     \label{\l__mikoslides_frame_label_str}
```

```
4841 }
```

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

```
4842 \def\itemize@level{outer}
4843 \def\itemize@outer{outer}
4844 \def\itemize@inner{inner}
4845 \renewcommand\newpage{\addtocounter{framenum}{1}}
4846 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4847 \renewenvironment{itemize}{
4848   \ifx\itemize@level\itemize@outer
4849     \def\itemize@label{$\rhd$}
4850   \fi
4851   \ifx\itemize@level\itemize@inner
4852     \def\itemize@label{$\scriptstyle\rhd$}
4853   \fi
4854   \begin{list}
4855     {\itemize@label}
4856     {\setlength{\labelsep}{.3em}
4857      \setlength{\labelwidth}{.5em}
4858      \setlength{\leftmargin}{1.5em}
4859     }
4860   \edef\itemize@level{\itemize@inner}
4861 }{
4862   \end{list}
4863 }
```

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

```
4864 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4865 }{
4866   \medskip\miko@slidelabel\end{mdframed}
4867 }
```

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

`\frametitle`

```
4868 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4869 }
```

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

EdN:21

`\pause` 21

```
4870 \bool_if:NT \c__mikoslides_notes_bool {
4871   \newcommand\pause{}
4872 }
```

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

`nomtext`

```
4873 \bool_if:NTF \c__mikoslides_notes_bool {
4874   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4875 }{
4876   \excludecomment{nomtext}
4877 }
```

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

nomgroup

```
4878 \bool_if:NTF \c__mikoslides_notes_bool {
4879   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4880 }{
4881   \excludecomment{nomgroup}
4882 }
```

ndefinition

```
4883 \bool_if:NTF \c__mikoslides_notes_bool {
4884   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
4885 }{
4886   \excludecomment{ndefinition}
4887 }
```

nassertion

```
4888 \bool_if:NTF \c__mikoslides_notes_bool {
4889   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
4890 }{
4891   \excludecomment{nassertion}
4892 }
```

nsproof

```
4893 \bool_if:NTF \c__mikoslides_notes_bool {
4894   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4895 }{
4896   \excludecomment{nproof}
4897 }
```

nexample

```
4898 \bool_if:NTF \c__mikoslides_notes_bool {
4899   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4900 }{
4901   \excludecomment{nexample}
4902 }
```

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

```
4903 \def\inputref@preskip{\smallskip}
4904 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
4905 \let\orig@inputref\inputref
4906 \def\inputref{\@ifstar\ninputref\orig@inputref}
4907 \newcommand\ninputref[2] [] {
4908   \bool_if:NT \c__mikoslides_notes_bool {
4909     \orig@inputref[#1]{#2}
4910   }
4911 }
```

(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>}`.

```

4912 \newlength{\slidelogoheight}
4913
4914 \bool_if:NTF \c__mikoslides_notes_bool {
4915   \setlength{\slidelogoheight}{.4cm}
4916 }{
4917   \setlength{\slidelogoheight}{1cm}
4918 }
4919 \newsavebox{\slidelogo}
4920 \sbox{\slidelogo}{\TeX}
4921 \newrobustcmd{\setslidelogo}[1]{
4922   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4923 }
```

(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.

```

4924 \def\source{Michael Kohlhase}% customize locally
4925 \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.

```

4926 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4927 \newsavebox{\cclogo}
4928 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4929 \newif\ifcchref\cchreffalse
4930 \AtBeginDocument{
4931   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4932 }
4933 \def\licensing{
4934   \ifcchref
4935     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4936   \else
4937     {\usebox{\cclogo}}
4938   \fi
4939 }
4940 \newrobustcmd{\setlicensing}[2][]{
4941   \def@url{#1}
4942   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4943   \ifx@url@empty
4944     \def\licensing{{\usebox{\cclogo}}}
4945   \else
4946     \def\licensing{
```

```

4947     \ifcchref
4948     \href{#1}{\usebox{\cclogo}}
4949     \else
4950     {\usebox{\cclogo}}
4951     \fi
4952   }
4953 \fi
4954 }

```

(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

```

4955 \newrobustcmd\miko@slidelabel{
4956   \vbox to \slidelogoheight{
4957     \vss\hbox to \slidewidth
4958     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4959   }
4960 }

```

(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.

```

4961 \def\Gin@mhrepos{}
4962 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4963 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4964 \newrobustcmd\frameimage[2][]{
4965   \stepcounter{slide}
4966   \bool_if:NT \c__mikoslides_frameimages_bool {
4967     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4968     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4969     \begin{center}
4970       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4971         \fbox{
4972           \ifx\Gin@ewidth\@empty
4973             \ifx\Gin@mhrepos\@empty
4974               \mhgraphics[width=\slidewidth,#1]{#2}
4975             \else
4976               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4977             \fi
4978           \else% \Gin@ewidth empty
4979             \ifx\Gin@mhrepos\@empty
4980               \mhgraphics[#1]{#2}
4981             \else
4982               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4983             \fi
4984           \fi% \Gin@ewidth empty
4985         }
4986       }{
4987         \ifx\Gin@ewidth\@empty

```

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

```

4988         \ifx\Gin@mhrepos\empty
4989             \mhgraphics[width=\slidewidth,#1]{#2}
4990         \else
4991             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4992         \fi
4993         \ifx\Gin@mhrepos\empty
4994             \mhgraphics[#1]{#2}
4995         \else
4996             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4997         \fi
4998     \fi% Gin@ewidth empty
4999 }
5000 \end{center}
5001 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5002 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
5003 }
5004 } % 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.

```

5005 \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

```

5006 \AddToHook{begindocument}{
5007     \definecolor{green}{rgb}{0,.5,0}
5008     \definecolor{purple}{cmyk}{.3,1,0,.17}
5009 }

```

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`.

```

5010 % \def\STpresent#1{\textcolor{blue}{#1}}
5011 \def\defemph#1{\textcolor{magenta}{#1}}
5012 \def\symrefemph#1{\textcolor{cyan}{#1}}
5013 \def\compemph#1{\textcolor{blue}{#1}}
5014 \def\titleemph#1{\textcolor{blue}{#1}}
5015 \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.

```

5016 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5017 \def\smalltextwarning{
5018     \pgfuseimage{miko@small@dbend}
5019     \xspace
5020 }
5021 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```



```

5022 \newrobustcmd\textwarning{
5023   \raisebox{-0.05cm}{\pgfuseimage{miko@dbend}}
5024   \xspace
5025 }
5026 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5027 \newrobustcmd\bigtextwarning{
5028   \raisebox{-0.05cm}{\pgfuseimage{miko@big@dbend}}
5029   \xspace
5030 }
(End definition for \textwarning. This function is documented on page ??.)
5031 \newrobustcmd\putgraphicsat[3]{
5032   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5033 }
5034 \newrobustcmd\putat[2]{
5035   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5036 }

```

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`.

```

5037 \bool_if:NT \c__mikoslides_sectocframes_bool {
5038   \str_if_eq:VnTF \__mikoslidestopsect{part}{
5039     \newcounter{chapter}\counterwithin*{section}{chapter}
5040   }{
5041     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
5042       \newcounter{chapter}\counterwithin*{section}{chapter}
5043     }
5044   }
5045 }

```

`\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
5046 \def\part@prefix{}
5047 \@ifpackageloaded{omdoc}{}{
5048   \str_case:VnF \__mikoslidestopsect {
5049     {part}{
5050       \int_set:Nn \l_document_structure_section_level_int {0}
5051       \def\thesection{\arabic{chapter}.\arabic{section}}
5052       \def\part@prefix{\arabic{chapter}.}
5053     }
5054     {chapter}{
5055       \int_set:Nn \l_document_structure_section_level_int {1}
5056       \def\thesection{\arabic{chapter}.\arabic{section}}
5057       \def\part@prefix{\arabic{chapter}.}
5058     }
5059   }{
5060     \int_set:Nn \l_document_structure_section_level_int {2}
5061     \def\part@prefix{}

```

```

5062 }
5063 }
5064
5065 \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

```

5066 \renewenvironment{omgroup}[2][]{
5067   \__document_structure_omgroup_args:n { #1 }
5068   \int_incr:N \l_document_structure_omgroup_level_int
5069   \int_incr:N \l_document_structure_section_level_int
5070   \bool_if:NT \c__mikoslides_sectocframes_bool {
5071     \stepcounter{slide}
5072     \begin{frame}[noframenumbering]
5073     \vfill\Large\centering
5074     \red{
5075       \ifcase\l_document_structure_section_level_int\or
5076         \stepcounter{part}
5077         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
5078         \def\currentsectionlevel{\omdoc@part@kw}
5079       \or
5080         \stepcounter{chapter}
5081         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5082         \def\currentsectionlevel{\omdoc@chapter@kw}
5083       \or
5084         \stepcounter{section}
5085         \def\__mikoslideslabel{\part@prefix\arabic{section}}
5086         \def\currentsectionlevel{\omdoc@section@kw}
5087       \or
5088         \stepcounter{subsection}
5089         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5090         \def\currentsectionlevel{\omdoc@subsection@kw}
5091       \or
5092         \stepcounter{subsubsection}
5093         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5094         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5095       \or
5096         \stepcounter{paragraph}
5097         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5098         \def\currentsectionlevel{\omdoc@paragraph@kw}
5099       \else
5100         \def\__mikoslideslabel{}
5101         \def\currentsectionlevel{\omdoc@paragraph@kw}
5102       \fi% end ifcase
5103       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
5104       \quad #2%
5105     }%
5106     \vfill%
5107     \end{frame}%
5108   }
5109   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5110 }{}
5111 }

```

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

```

5112 \def\inserttheorembodyfont{\normalfont}
5113 %\bool_if:NF \c__mikoslides_notes_bool {
5114 % \defbeamertemplate{theorem begin}{miko}
5115 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5116 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5117 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5118 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5119 % \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.

```

5120 % \expandafter\def\csname Parent2\endcsname{}
5121 %}
5122
5123 \AddToHook{begindocument}{% this does not work for some reasons
5124 \setbeamertemplate{theorems}[ams style]
5125 }
5126 \bool_if:NT \c__mikoslides_notes_bool {
5127 \renewenvironment{columns}[1][{}]{%
5128 \par\noindent%
5129 \begin{minipage}%
5130 \slidewidth\centering\leavevmode%
5131 }{}%
5132 \end{minipage}\par\noindent%
5133 }%
5134 \newsavebox\columnbox%
5135 \renewenvironment<>{column}[2][{}]{%
5136 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5137 }{}%
5138 \end{minipage}\end{lrbox}\usebox\columnbox%
5139 }%
5140 }
5141 \bool_if:NTF \c__mikoslides_noproblems_bool {
5142 \newenvironment{problems}{}{}
5143 }{
5144 \excludecomment{problems}
5145 }

```

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.

```

5146 \gdef\printexcursions{}
5147 \newcommand\excursionref[2]{% label, text
5148 \bool_if:NT \c__mikoslides_notes_bool {

```

```

5149 \begin{sparagraph}[title=Excursion]
5150 #2 \sref[fallback=the appendix]{#1}.
5151 \end{sparagraph}
5152 }
5153 }
5154 \newcommand\activate@excursion[2][]{
5155 \gappto\printexcursions{\inputref[#1]{#2}}
5156 }
5157 \newcommand\excursion[4][]{% repos, label, path, text
5158 \bool_if:NT \c__mikoslides_notes_bool {
5159 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5160 }
5161 }

```

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

\excursiongroup

```

5162 \keys_define:nn{mikoslides / excursiongroup }{
5163 id .str_set_x:N = \l__mikoslides_excursion_id_str,
5164 intro .tl_set:N = \l__mikoslides_excursion_intro_tl,
5165 mhrepos .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
5166 }
5167 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
5168 \tl_clear:N \l__mikoslides_excursion_intro_tl
5169 \str_clear:N \l__mikoslides_excursion_id_str
5170 \str_clear:N \l__mikoslides_excursion_mhrepos_str
5171 \keys_set:nn {mikoslides / excursiongroup }{ #1 }
5172 }
5173 \newcommand\excursiongroup[1][]{
5174 \__mikoslides_excursion_args:n{ #1 }
5175 \ifdefempty\printexcursions{}% only if there are excursions
5176 {\begin{note}
5177 \begin{omgroup}[#1]{Excursions}%
5178 \ifdefempty\l__mikoslides_excursion_intro_tl{{
5179 \inputref[\l__mikoslides_excursion_mhrepos_str]{
5180 \l__mikoslides_excursion_intro_tl
5181 }
5182 }
5183 \printexcursions%
5184 \end{omgroup}
5185 \end{note}}
5186 }
5187 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5188 \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.

```
5189 <*package>
5190 <@@=problems>
5191 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5192 \RequirePackage{l3keys2e,expl-keystr-compatible}
5193
5194 \keys_define:nn { problem / pkg }{
5195   notes      .default:n    = { true },
5196   notes      .bool_set:N   = \c__problems_notes_bool,
5197   gnotes     .default:n    = { true },
5198   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5199   hints      .default:n    = { true },
5200   hints      .bool_set:N   = \c__problems_hints_bool,
5201   solutions  .default:n    = { true },
5202   solutions  .bool_set:N   = \c__problems_solutions_bool,
5203   pts        .default:n    = { true },
5204   pts        .bool_set:N   = \c__problems_pts_bool,
5205   min        .default:n    = { true },
5206   min        .bool_set:N   = \c__problems_min_bool,
5207   boxed      .default:n    = { true },
5208   boxed      .bool_set:N   = \c__problems_boxed_bool,
5209   unknown    .code:n       = {}
5210 }
5211 \def\solutionstrue{
5212   \bool_set_true:N \c__problems_solutions_bool
5213 }
5214 \def\solutionsfalse{
5215   \bool_set_false:N \c__problems_solutions_bool
5216 }
5217
5218 \ProcessKeysOptions{ problem / pkg }
```

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

```

5219 \RequirePackage{stex-compatibility}
5220 \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.

```

5221 \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.

```

5222 \def\prob@problem@kw{Problem}
5223 \def\prob@solution@kw{Solution}
5224 \def\prob@hint@kw{Hint}
5225 \def\prob@note@kw{Note}
5226 \def\prob@gnote@kw{Grading}
5227 \def\prob@pt@kw{pt}
5228 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

5229 \@ifpackageloaded{babel}{
5230   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5231   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5232     \input{problem-ngerman.ldf}
5233   }
5234   \clist_if_in:NnT \l_tmpa_clist {finnish}{
5235     \input{problem-finnish.ldf}
5236   }
5237   \clist_if_in:NnT \l_tmpa_clist {french}{
5238     \input{problem-french.ldf}
5239   }
5240   \clist_if_in:NnT \l_tmpa_clist {russian}{
5241     \input{problem-russian.ldf}
5242   }
5243 }{}

```

40.2 Problems and Solutions

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

```

5244 \keys_define:nn{ problem / problem }{
5245   id      .str_set:x:N = \l__problems_prob_id_str,
5246   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5247   min     .tl_set:N    = \l__problems_prob_min_tl,
5248   title   .tl_set:N    = \l__problems_prob_title_tl,
5249   refnum  .int_set:N   = \l__problems_prob_refnum_int
5250 }
5251 \cs_new_protected:Nn \__problems_prob_args:n {
5252   \str_clear:N \l__problems_prob_id_str
5253   \tl_clear:N \l__problems_prob_pts_tl
5254   \tl_clear:N \l__problems_prob_min_tl
5255   \tl_clear:N \l__problems_prob_title_tl

```

```

5256 \int_zero_new:N \l__problems_prob_refnum_int
5257 \keys_set:nn { problem / problem }{ #1 }
5258 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5259   \let\l__problems_inclprob_refnum_int\undefined
5260 }
5261 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5262 \newcounter{problem}
5263 \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.

```

5264 \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

```

5265 \newcommand\prob@number{
5266   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5267     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5268   }{
5269     \int_if_exist:NTF \l__problems_prob_refnum_int {
5270       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5271     }{
5272       \prob@label\theproblem
5273     }
5274   }
5275 }

```

(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.

```

5276 \newcommand\prob@title[3]{%
5277   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5278     #2 \l__problems_inclprob_title_tl #3
5279   }{
5280     \tl_if_exist:NTF \l__problems_prob_title_tl {
5281       #2 \l__problems_prob_title_tl #3
5282     }{
5283       #1
5284     }
5285   }
5286 }

```

(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.

```

5287 \def\prob@heading{
5288   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5289   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5290 }

```

(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`

```

5291 \newenvironment{problem}[1][1]{
5292   \__problems_prob_args:n{#1}%\sref@target%
5293   \@in@omtexttrue% we are in a statement (for inline definitions)
5294   \stepcounter{problem}\record@problem
5295   \def\current@section@level{\prob@problem@kw}
5296   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5297 }%
5298 {\smallskip}
5299 \bool_if:NT \c__problems_boxed_bool {
5300   \surroundwithmdframed{problem}
5301 }

```

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

```

5302 \def\record@problem{
5303   \protected@write\@auxout{}
5304   {
5305     \string\@problem{\prob@number}
5306     {
5307       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5308         \l__problems_inclprob_pts_tl
5309       }{
5310         \l__problems_prob_pts_tl
5311       }
5312     }%
5313     {
5314       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5315         \l__problems_inclprob_min_tl
5316       }{
5317         \l__problems_prob_min_tl
5318       }
5319     }
5320   }
5321 }

```

(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).

```

5322 \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.

```

5323 \keys_define:nn { problem / solution }{
5324   id          .str_set_x:N = \l__problems_solution_id_str ,
5325   for         .tl_set:N    = \l__problems_solution_for_tl ,
5326   height      .dim_set:N   = \l__problems_solution_height_dim ,
5327   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5328   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5329   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5330 }
5331 \cs_new_protected:Nn \__problems_solution_args:n {
5332   \str_clear:N \l__problems_solution_id_str
5333   \tl_clear:N \l__problems_solution_for_tl
5334   \tl_clear:N \l__problems_solution_srccite_tl
5335   \clist_clear:N \l__problems_solution_creators_clist
5336   \clist_clear:N \l__problems_solution_contributors_clist
5337   \dim_zero:N \l__problems_solution_height_dim
5338   \keys_set:nn { problem / solution }{ #1 }
5339 }

```

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

```

5340 \newcommand\@startsolution[1][ ]{
5341   \__problems_solution_args:n { #1 }
5342   \@in@omtexttrue% we are in a statement.
5343   \bool_if:NF \c__problems_boxed_bool { \hrule }
5344   \smallskip\noindent
5345   {\textbf\prob@solution@kw : \enspace}
5346   \begin{small}
5347   \def\current@section@level{\prob@solution@kw}
5348   \ignorespacesandpars
5349 }

```

\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.

```

5350 \newcommand\startsolutions{
5351   \specialcomment{solution}{\@startsolution}{
5352     \bool_if:NF \c__problems_boxed_bool {
5353       \hrule\medskip
5354     }
5355     \end{small}%
5356   }
5357   \bool_if:NT \c__problems_boxed_bool {
5358     \surroundwithmdframed{solution}
5359   }
5360 }

```

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

\stopsolutions

```

5361 \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.

```

5362 \bool_if:NTF \c_problems_solutions_bool {
5363   \startsolutions
5364 }{
5365   \stopsolutions
5366 }

```

exnote

```

5377 \bool_if:NTF \c_problems_notes_bool {
5378   \newenvironment{exnote}[1][]{
5379     \par\smallskip\hrule\smallskip
5380     \noindent\textbf{\prob@note@kw : }\small
5381   }{
5382     \smallskip\hrule
5383   }
5384 }{
5385   \excludecomment{exnote}
5386 }

```

hint

```

5377 \bool_if:NTF \c_problems_notes_bool {
5378   \newenvironment{hint}[1][]{
5379     \par\smallskip\hrule\smallskip
5380     \noindent\textbf{\prob@hint@kw :~ }\small
5381   }{
5382     \smallskip\hrule
5383   }
5384   \newenvironment{exhint}[1][]{
5385     \par\smallskip\hrule\smallskip
5386     \noindent\textbf{\prob@hint@kw :~ }\small
5387   }{
5388     \smallskip\hrule
5389   }
5390 }{
5391   \excludecomment{hint}
5392   \excludecomment{exhint}
5393 }

```

gnote

```

5394 \bool_if:NTF \c_problems_notes_bool {
5395   \newenvironment{gnote}[1][]{
5396     \par\smallskip\hrule\smallskip
5397     \noindent\textbf{\prob@gnote@kw : }\small
5398   }{
5399     \smallskip\hrule
5400   }
5401 }{
5402   \excludecomment{gnote}
5403 }

```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
5404 \newenvironment{mcb}{
5405   \begin{enumerate}
5406 }{
5407   \end{enumerate}
5408 }
```

we define the keys for the mcc macro

```
5409 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5410   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5411     \bool_set_true:N #1
5412   }{
5413     \bool_set_false:N #1
5414   }
5415 }
5416 \keys_define:nn { problem / mcc }{
5417   id          .str_set:x:N = \l__problems_mcc_id_str ,
5418   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5419   T           .default:n    = { true } ,
5420   T           .bool_set:N    = \l__problems_mcc_t_bool ,
5421   F           .default:n    = { true } ,
5422   F           .bool_set:N    = \l__problems_mcc_f_bool ,
5423   Ttext       .code:n       = {
5424     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5425   } ,
5426   Ftext       .code:n       = {
5427     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5428   }
5429 }
5430 \cs_new_protected:Nn \l__problems_mcc_args:n {
5431   \str_clear:N \l__problems_mcc_id_str
5432   \tl_clear:N \l__problems_mcc_feedback_tl
5433   \bool_set_true:N \l__problems_mcc_t_bool
5434   \bool_set_true:N \l__problems_mcc_f_bool
5435   \bool_set_true:N \l__problems_mcc_Ttext_bool
5436   \bool_set_false:N \l__problems_mcc_Ftext_bool
5437   \keys_set:nn { problem / mcc }{ #1 }
5438 }
```

\mcc

```
5439 \newcommand\mcc[2][] {
5440   \l__problems_mcc_args:n{ #1 }
5441   \item #2
5442   \bool_if:NT \c__problems_solutions_bool {
5443     \
5444     \bool_if:NT \l__problems_mcc_t_bool {
5445       % TODO!
5446       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5447     }
5448     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

5449      % TODO!
5450      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5451    }
5452    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5453      !
5454    }{
5455      \l__problems_mcc_feedback_tl
5456    }
5457  }
5458 } %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.

```

5459
5460 \keys_define:nn{ problem / inclproblem }{
5461   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5462   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5463   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5464   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5465   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5466   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5467 }
5468 \cs_new_protected:Nn \__problems_inclprob_args:n {
5469   % \str_clear:N \l__problems_prob_id_str
5470   \tl_clear:N \l__problems_inclprob_pts_tl
5471   \tl_clear:N \l__problems_inclprob_min_tl
5472   \tl_clear:N \l__problems_inclprob_title_tl
5473   \int_zero_new:N \l__problems_inclprob_refnum_int
5474   \str_clear:N \l__problems_inclprob_mhrepos_str
5475   \keys_set:nn { problem / inclproblem }{ #1 }
5476   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5477     \let\l__problems_inclprob_pts_tl\undefined
5478   }
5479   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5480     \let\l__problems_inclprob_min_tl\undefined
5481   }
5482   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5483     \let\l__problems_inclprob_title_tl\undefined
5484   }
5485   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5486     \let\l__problems_inclprob_refnum_int\undefined
5487   }
5488 }
5489
5490 \cs_new_protected:Nn \__problems_inclprob_clear: {
5491   % \str_clear:N \l__problems_prob_id_str
5492   \let\l__problems_inclprob_pts_tl\undefined
5493   \let\l__problems_inclprob_min_tl\undefined

```

```

5494 \let\l__problems_inclprob_title_tl\undefined
5495 \let\l__problems_inclprob_refnum_int\undefined
5496 \let\l__problems_inclprob_mhrepos_str\undefined
5497 }
5498
5499 \newcommand\includeproblem[2][ ]{
5500   \__problems_inclprob_args:n{ #1 }
5501   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5502     \input{#2}
5503   }{
5504     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5505       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5506     }
5507   }
5508   \__problems_inclprob_clear:
5509 }

```

(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.

```

5510 \AddToHook{enddocument}{
5511   \bool_if:NT \c__problems_pts_bool {
5512     \message{Total:~\arabic{pts}~points}
5513   }
5514   \bool_if:NT \c__problems_min_bool {
5515     \message{Total:~\arabic{min}~minutes}
5516   }
5517 }

```

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

```

5518 \def\pts#1{
5519   \bool_if:NT \c__problems_pts_bool {
5520     \marginpar{#1~\prob@pt@kw}
5521   }
5522 }
5523 \def\min#1{
5524   \bool_if:NT \c__problems_min_bool {
5525     \marginpar{#1~\prob@min@kw}
5526   }
5527 }

```

\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.

```

5528 \newcounter{pts}
5529 \def\show@pts{
5530   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5531     \bool_if:NT \c__problems_pts_bool {
5532       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5533       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5534     }
5535   }{
5536     \tl_if_exist:NT \l__problems_prob_pts_tl {
5537       \bool_if:NT \c__problems_pts_bool {
5538         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5539         \addtocounter{pts}{\l__problems_prob_pts_tl}
5540       }
5541     }
5542   }
5543 }

```

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

\show@min

```

5544 \newcounter{min}
5545 \def\show@min{
5546   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5547     \bool_if:NT \c__problems_min_bool {
5548       \marginpar{\l__problems_inclprob_pts_tl;min}
5549       \addtocounter{min}{\l__problems_inclprob_min_tl}
5550     }
5551   }{
5552     \tl_if_exist:NT \l__problems_prob_min_tl {
5553       \bool_if:NT \c__problems_min_bool {
5554         \marginpar{\l__problems_prob_min_tl;min}
5555         \addtocounter{min}{\l__problems_prob_min_tl}
5556       }
5557     }
5558   }
5559 }
5560 \</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.

```
5561 <@@=hwexam>
5562 <*cls>
5563 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5564 \RequirePackage{l3keys2e,expl-keystr-compatible}
5565 \DeclareOption*{
5566   \PassOptionsToClass{\CurrentOption}{omdoc}
5567   \PassOptionsToPackage{\CurrentOption}{stex}
5568   \PassOptionsToPackage{\CurrentOption}{hwexam}
5569   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5570 }
5571 \ProcessOptions
```

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

```
5572 \LoadClass{omdoc}
5573 \RequirePackage{stex}
5574 \RequirePackage{hwexam}
5575 \RequirePackage{tikzinput}
5576 \RequirePackage{graphicx}
5577 \RequirePackage{a4wide}
5578 \RequirePackage{amssymb}
5579 \RequirePackage{amstext}
5580 \RequirePackage{amsmath}
```

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

```

5581 \newcommand\assig@default@type{\hwexam@assignment@kw}
5582 \def\document@hwexamtype{\assig@default@type}
5583 <@@=document_structure>
5584 \keys_define:nn { document-structure / document }{
5585 id .str_set_x:N = \c_document_structure_document_id_str,
5586 hwexamtype .tl_set:N = \document@hwexamtype
5587 }
5588 <@@=hwexam>
5589 </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.

```
5590 \langle *package \rangle
5591 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5592 \RequirePackage{l3keys2e,expl-keystr-compat}
5593
5594 \newif\iftest\testfalse
5595 \DeclareOption{test}{\testtrue}
5596 \newif\ifmultiple\multiplefalse
5597 \DeclareOption{multiple}{\multipletrue}
5598 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5599 \ProcessOptions
```

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

```
5600 \RequirePackage{keyval}[1997/11/10]
5601 \RequirePackage{problem}
```

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

```
5602 \newcommand\hwexam@assignment@kw{Assignment}
5603 \newcommand\hwexam@given@kw{Given}
5604 \newcommand\hwexam@due@kw{Due}
5605 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
5606 blank~for~extra~space}%
5607 \newcommand\correction@probs@kw{prob.}%
5608 \newcommand\correction@pts@kw{total}%
5609 \newcommand\correction@reached@kw{reached}%
5610 \newcommand\correction@sum@kw{Sum}%
5611 \newcommand\correction@grade@kw{grade}%
5612 \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

```

5613 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5614
5615 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5616 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5617   \input{hwexam-ngerman.ldf}
5618 }
5619 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5620   \input{hwexam-finnish.ldf}
5621 }
5622 \clist_if_in:NnT \l_tmpa_clist {french}{
5623   \input{hwexam-french.ldf}
5624 }
5625 \clist_if_in:NnT \l_tmpa_clist {russian}{
5626   \input{hwexam-russian.ldf}
5627 }

```

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.

```

5628 \newcounter{assignment}
5629 \numberproblemsin{assignment}
5630 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5631 \keys_define:nn { hwexam / assignment } {
5632   id .str_set:N = \l__hwexam_assign_id_str,
5633   number .int_set:N = \l__hwexam_assign_number_int,
5634   title .tl_set:N = \l__hwexam_assign_title_tl,
5635   type .tl_set:N = \l__hwexam_assign_type_tl,
5636   given .tl_set:N = \l__hwexam_assign_given_tl,
5637   due .tl_set:N = \l__hwexam_assign_due_tl,
5638   loadmodules .code:n = {
5639     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5640   }
5641 }
5642 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5643   \str_clear:N \l__hwexam_assign_id_str
5644   \int_set:Nn \l__hwexam_assign_number_int {-1}
5645   \tl_clear:N \l__hwexam_assign_title_tl
5646   \tl_clear:N \l__hwexam_assign_type_tl
5647   \tl_clear:N \l__hwexam_assign_given_tl
5648   \tl_clear:N \l__hwexam_assign_due_tl
5649   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5650   \keys_set:nn { hwexam / assignment }{ #1 }
5651 }

```

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.

```

5652 \newcommand\given@due[2]{
5653 \bool_lazy_all:nF {
5654 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5655 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5656 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5657 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5658 }{ #1 }
5659
5660 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5661 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5662 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5663 }
5664 }{
5665 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5666 }
5667
5668 \bool_lazy_or:nnF {
5669 \bool_lazy_and_p:nn {
5670 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5671 }{
5672 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5673 }
5674 }{
5675 \bool_lazy_and_p:nn {
5676 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5677 }{
5678 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5679 }
5680 }{ ,~ }
5681
5682 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5683 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5684 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5685 }
5686 }{
5687 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5688 }
5689
5690 \bool_lazy_all:nF {
5691 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5692 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5693 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5694 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5695 }{ #2 }
5696 }

```

`\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.

```

5697 \newcommand\assignment@title[3]{

```

```

5698 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5699 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5700 #1
5701 }{
5702 #2\l__hwexam_assign_title_tl#3
5703 }
5704 }{
5705 #2\l__hwexam_inclassassign_title_tl#3
5706 }
5707 }

```

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

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

```

5708 \newcommand\assignment@number{
5709 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5710 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5711 \int_use:N \l__hwexam_assign_number_int
5712 }
5713 }{
5714 \int_use:N \l__hwexam_inclassassign_number_int
5715 }
5716 }

```

(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.

```

5717 \newenvironment{assignment}[1][ ]{
5718 \__hwexam_assignment_args:n { #1 }
5719 %\sref@target
5720 \let\__hwexamnum\l__hwexam_assign_number_int
5721 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5722 \stepcounter{assignment}
5723 }{
5724 \setcounter{assignment}{\int_use:N\__hwexamnum}
5725 }
5726 \setcounter{problem}{0}
5727 \def\current@section@level{\document@hwexamtype}
5728 %\sref@label@id{\document@hwexamtype \thesection}
5729 \begin{@assignment}
5730 }{
5731 \end{@assignment}
5732 }

```

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

```

5733 \def\__hwexasstitle{
5734 \protect\document@hwexamtype~\arabic{assignment}
5735 \assignment@title{}\;{} \; -- \given@due{}\}
5736 }

```

```

5737 \ifmultiple
5738 \newenvironment{@assignment}{
5739 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5740 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5741 }{
5742 \begin{omgroup}{\__hwexasstitle}
5743 }
5744 }{
5745 \end{omgroup}
5746 }

```

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

```

5747 \else
5748 \newenvironment{@assignment}{
5749 \begin{center}\bf
5750 \Large\@title\strut\
5751 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}\
5752 \large\given@due{--\;}{\;--}
5753 \end{center}
5754 }{}
5755 \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.

```

5756 \keys_define:nn { hwexam / inclassignment } {
5757 %id .str_set_x:N = \l__hwexam_assign_id_str,
5758 number .int_set:N = \l__hwexam_inclassign_number_int,
5759 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5760 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5761 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5762 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5763 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5764 }
5765 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5766 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5767 \tl_clear:N \l__hwexam_inclassign_title_tl
5768 \tl_clear:N \l__hwexam_inclassign_type_tl
5769 \tl_clear:N \l__hwexam_inclassign_given_tl
5770 \tl_clear:N \l__hwexam_inclassign_due_tl
5771 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5772 \keys_set:nn { hwexam / inclassignment }{ #1 }
5773 }
5774 \__hwexam_inclassignment_args:n {}
5775
5776 \newcommand\inputassignment[2][ ]{
5777 \__hwexam_inclassignment_args:n { #1 }
5778 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5779 \input{#2}
5780 }{
5781 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5782 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5783 }
5784 }
5785 \__hwexam_inclasssignment_args:n {}
5786 }
5787 \newcommand\includeassignment[2][ ]{
5788 \newpage
5789 \inputassignment[#1]{#2}
5790 }

```

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

42.4 Typesetting Exams

\quizheading

```

5791 \ExplSyntaxOff
5792 \newcommand\quizheading[1]{%
5793 \def\@tas{#1}%
5794 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5795 \ifx\@tas\empty\else%
5796 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5797 \fi%
5798 }
5799 \ExplSyntaxOn

```

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

\testheading

```

5800 \keys_define:nn { hwexam / testheading } {
5801 min .tl_set:N = \l__hwexam_testheading_min_tl,
5802 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5803 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5804 }
5805 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5806 \tl_clear:N \l__hwexam_testheading_min_tl
5807 \tl_clear:N \l__hwexam_testheading_duration_tl
5808 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5809 \keys_set:nn { hwexam / testheading }{ #1 }
5810 }
5811 \newenvironment{testheading}[1][ ]{
5812 \__hwexam_testheading_args:n{ #1 }
5813 \noindent\large{Name:~\hfill
5814 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5815 \begin{center}
5816 \Large\textbf{\@title}\[1ex]
5817 \large\@date\[3ex]
5818 \end{center}
5819 \textbf{You~have~
5820 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5821 {\l__hwexam_testheading_min_tl}~minutes
5822 }{
5823 {\l__hwexam_testheading_duration_tl}
5824 }~

```

```

5825 (sharp)~for~the~test
5826 };\
5827 Write~the~solutions~to~the~sheet.
5828 \par\noindent
5829 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5830 \advance\check@time by -\theassignment@totalmin
5831 The~estimated~time~for~solving~this~exam~is~
5832 {\theassignment@totalmin}-minutes,~
5833 leaving~you~{\the\check@time}-minutes~for~revising~
5834 your~exam.
5835
5836 \par\noindent
5837 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5838 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5839 You~can~reach~{\theassignment@totalpts}-points~if~you~
5840 solve~all~problems.~You~will~only~need~
5841 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5842 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5843 \vfill
5844 \begin{center}
5845 {
5846 \Large\em You~have~ample~time,~so~take~it~slow~
5847 and~avoid~rushing~to~mistakes!\}[2ex]
5848 Different~problems~test~different~skills~and~
5849 knowledge,~so~do~not~get~stuck~on~one~problem.
5850 }
5851 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5852 \end{center}
5853 }{
5854 \newpage
5855 }

```

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

\testspace

```

5856 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5857 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5858 \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.

```

5859 <@=problems>
5860 \renewcommand\@problem[3]{
5861 \stepcounter{assignment@probs}
5862 \def\__problemspts{#2}

```

```

5863 \ifx\__problemspts\@empty\else
5864 \addtocounter{assignment@totalpts}{#2}
5865 \fi
5866 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5867 \xdef\correction@probs{\correction@probs & #1}%
5868 \xdef\correction@pts{\correction@pts & #2}
5869 \xdef\correction@reached{\correction@reached & }
5870 }
5871 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5872 \newcounter{assignment@probs}
5873 \newcounter{assignment@totalpts}
5874 \newcounter{assignment@totalmin}
5875 \def\correction@probs{\correction@probs@kw}%
5876 \def\correction@pts{\correction@pts@kw}%
5877 \def\correction@reached{\correction@reached@kw}%
5878 \def\after@correction@table{}%
5879 \stepcounter{assignment@probs}
5880 \newcommand\correction@table{
5881 \resizebox{\textwidth}{!}{%
5882 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5883 &\multicolumn{\theassignment@probs}{c|}||%|
5884 {\footnotesize\correction@forgrading@kw} &\\ \hline
5885 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5886 \correction@pts & \theassignment@totalpts & \\ \hline
5887 \correction@reached & & \[.7cm]\hline
5888 \end{tabular}}
5889 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5890 \end{package}

```

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

42.5 Leftovers

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

here we define the logos that characterize the assignment

```

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

```

```

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

```