

The sTeX3 Package *

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

2022-02-12

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 2022-02-12)

Contents

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

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

21 document-structure: Semantic Markup for Open Mathematical Documents in \LaTeX	53
21.1 Introduction	53
21.2 The User Interface	54
21.2.1 Package and Class Options	54
21.2.2 Document Structure	54
21.2.3 Ignoring Inputs	56
21.2.4 Structure Sharing	56
21.2.5 Global Variables	56
21.2.6 Colors	57
21.3 Limitations	57
22 NotesSlides – Slides and Course Notes	58
22.1 Introduction	58
22.2 The User Interface	58
22.2.1 Package Options	58
22.2.2 Notes and Slides	59
22.2.3 Header and Footer Lines of the Slides	60
22.2.4 Frame Images	60
22.2.5 Colors and Highlighting	61
22.2.6 Front Matter, Titles, etc.	61
22.2.7 Excursions	61
22.2.8 Miscellaneous	62
22.3 Limitations	62
23 problem.sty: An Infrastructure for formatting Problems	63
23.1 Introduction	63
23.2 The User Interface	63
23.2.1 Package Options	63
23.2.2 Problems and Solutions	64
23.2.3 Multiple Choice Blocks	65
23.2.4 Including Problems	65
23.2.5 Reporting Metadata	65
23.3 Limitations	65
24 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	67
24.1 Introduction	68
24.2 The User Interface	68
24.2.1 Package and Class Options	68
24.2.2 Assignments	68
24.2.3 Typesetting Exams	68
24.2.4 Including Assignments	69
24.3 Limitations	69
IV Implementation	71

25	STeX-Basics Implementation	72
25.1	The STeXDocument Class	72
25.2	Preliminaries	72
25.3	Messages and logging	73
25.4	Persistence	74
25.5	HTML Annotations	74
25.6	Languages	77
25.7	Activating/Deactivating Macros	78
26	STeX-MathHub Implementation	80
26.1	Generic Path Handling	80
26.2	PWD and kpsewhich	82
26.3	File Hooks and Tracking	83
26.4	MathHub Repositories	84
27	STeX-References Implementation	92
27.1	Document URIs and URLs	92
27.2	Setting Reference Targets	94
27.3	Using References	95
28	STeX-Modules Implementation	98
28.1	The module environment	101
28.2	Invoking modules	107
29	STeX-Module Inheritance Implementation	109
29.1	SMS Mode	109
29.2	Inheritance	113
30	STeX-Symbols Implementation	118
30.1	Symbol Declarations	118
30.2	Notations	125
31	STeX-Terms Implementation	134
31.1	Symbol Invocations	134
31.2	Terms	137
31.3	Notation Components	143
32	STeX-Structural Features Implementation	146
32.1	Imports with modification	146
32.2	The feature environment	153
32.3	Features	155
33	STeX-Statements Implementation	160
33.1	Definitions	160
33.2	Assertions	164
33.3	Examples	166
33.4	Logical Paragraphs	168

34 The Implementation	171
34.1 Package Options	171
34.2 Proofs	171
34.3 Justifications	177
35 \LaTeX-Others Implementation	179
36 \LaTeX-Metatheory Implementation	180
37 Tikzinput Implementation	183
38 document-structure.sty Implementation	185
38.1 The document-structure Class	185
38.2 Class Options	185
38.3 Beefing up the <code>document</code> environment	186
38.4 Implementation: document-structure Package	186
38.5 Package Options	186
38.6 Document Structure	188
38.7 Front and Backmatter	191
38.8 Global Variables	193
39 NotesSlides – Implementation	194
39.1 Class and Package Options	194
39.2 Notes and Slides	196
39.3 Header and Footer Lines	200
39.4 Frame Images	201
39.5 Colors and Highlighting	202
39.6 Sectioning	203
39.7 Excursions	206
40 The Implementation	207
40.1 Package Options	207
40.2 Problems and Solutions	208
40.3 Multiple Choice Blocks	214
40.4 Including Problems	215
40.5 Reporting Metadata	216
41 Implementation: The hwexam Class	218
41.1 Class Options	218
42 Implementation: The hwexam Package	220
42.1 Package Options	220
42.2 Assignments	221
42.3 Including Assignments	224
42.4 Typesetting Exams	225
42.5 Leftovers	227

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

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

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using Semantic Macros

TODO

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

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

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

`url`: The URL that is formed as a basis for *external references*, see (TODO),

`dependencies`: All archives that this archive depends on. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

Chapter 5

Creating New Modules and Symbols

TODO

5.1 Advanced Structuring Mechanisms

Given modules:

Example 1

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

Module 5.1.1[magma]

Module 5.1.2[monoid]

Module 5.1.3[group]

We can form a module for *rings* by “cloning” an instance of **group** (for addition) and **monoid** (for multiplication), respectively, and “glueing them together” to ensure they share the same universe:

Example 2

```
\begin{module}{ring}
\begin{copymodule}{group}{addition}
\renamedekl[name=universe]{universe}{runiverse}
\renamedekl[name=plus]{operation}{rplus}
\renamedekl[name=zero]{unit}{rzero}
\renamedekl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{runiverse}
\renamedekl[name=times]{operation}{rtimes}
\renamedekl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\compl}

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

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

TODO: explain donotclone

Example 3

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

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

Module 5.1.5[int]

5.2 Primitive Symbols (The sTeX Metatheory)

Chapter 6

TeX Statements (Definitions, Theorems, Examples, ...)

Chapter 7

Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 4

```
\symdecl[args=2]{mult}
\notation{mult}{#1 #2}
 $\mult{a}{b}$ 
```

ab

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

```
\symdef[args=2]{mult}{#1 #2}
```

Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$:

Example 5

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
 $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ 
```

$a \cdot b$ and $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed⁴.

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 6

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

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

.

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

Example 7

```
\mult[\comp{Multiplying}]* $\mult{a}{b}$ [ again by  $b$  yields ...]
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 8

```
\symdecl[ args=2]{forevery}
\forevery* [2]{The proposition  $P$ }[ $\comp{holds for every}$ ][1]{ $x \in A$ }
```

The proposition P holds for every $x \in A$

⁴EdNOTE: TODO

When using `*[n]`, after reading the provided (n th) argument, the “argument counter” automatically continues where we left off, so the `*[1]` in the above example can be omitted.

For a macro with `arity > 0`, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

Example 9

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

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

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

Example 10

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

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

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

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

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

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

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

Other Argument Types

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

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

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

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

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

b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDOC and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 11

```
\symdef[ args=a]{mult}{#1}{#1 \comp\cdot #2}
$\mult{a,b,c,{d^e},f}$
```

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 12

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

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

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

5 6

⁵EDNOTE: what about e.g. $\int \int \int f(x,y,z) dx dy dz$?

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

Precedences

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

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

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

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

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

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

For example:

Example 13

```
\notation[prec=100]{plus}{#1 \comp{+} #2}
\notation[prec=50]{times}{#1 \comp{\cdot} #2}
 $\plus{a}{\times{b}{c}}$  and  $\times{a}{\plus{b}{c}}$ 
```

$a + b \cdot c$ and $a \cdot (b + c)$

8.1.2 Archives and Imports

Namespaces

Ideally, \TeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

Unfortunately, \TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that \TeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{module}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix¹.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\importmodule`:

- `\importmodule{Foo}` outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

The module `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

Since this is less compatible with a modular development, using full URIs directly is discouraged.

¹which is internally attached to the module name instead, but a user need not worry about that.

Part II

Documentation

Chapter 9

sTeX-Basics

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

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

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

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

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

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

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

9.1 Macros and Environments

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

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

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

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

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

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

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

Annotates the HTML generated by `<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:NTF</code>	\star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

10.1.2 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

12.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

Test 5

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

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

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

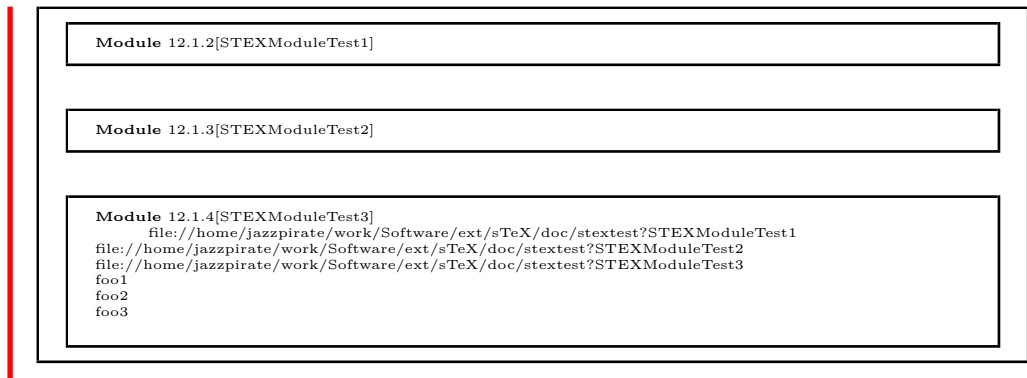
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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



`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

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

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

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

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

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

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

Test 9

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

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

Module 13.1.4[UseTest1]

Module 13.1.5[UseTest2]

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

Module 13.1.6[UseTest3]

Meaning: >undefined<

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

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

Test 10

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

Circular dependencies:

```
Module 13.1.7[CircDep1]
  >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
  >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo//circular2?Circular2?fooB}<
```

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

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

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

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

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.
- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

- (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`. That module should lie directly in the namespace of the archive.
- (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\stex_import_require_module:nnnn` `{<ns>} {<archive-ID>} {<path>} {<name>}`

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

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

Chapter 14

TEX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 14.1.2[NotationTest]

`\symdef`

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

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

Test 13

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

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

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

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

Test 14

```

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

```

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

Test 15

```

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

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

18.2.5 Proof End Markers

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

`\sproofend`

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

`\sProofEndSymbol`

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

18.2.6 Configuration of the Presentation

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

EdN:8

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

21.1 Introduction

\S TeX is a version of 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). The package supports direct translation to the OMDOC format [Koh06]

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

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

source and the formatter does the copying during document formatting/presentation.⁹

21.2 The User Interface

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

21.2.1 Package and Class Options

The `document-structure` 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 `document-structure` package accepts the same except the first two.

21.2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble ² . This can be used to give metadata about the
<code>id</code>	document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code>
<code>omgroup</code>	element resulting from the L ^A T _E XML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the L ^A T _E X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com-
	mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.
	Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for
	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>id</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>creators</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>contributors</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>short</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>
<code>loadmodules</code>	key it needs no value. For instance we would have

```

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

```

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.

`blindomgroup`

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

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

21.2.3 Ignoring Inputs

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

`\prematurestop` 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.

`\afterprematurestop` `\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.

`\setSGvar`
`\useSGvar`
`\ifSGvar`

With `\ifSGvar` we can test for the contents of a global variable: the macro call

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

`\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable `⟨vname⟩`, only if (after expansion) it is equal to `⟨val⟩`, the conditional text `⟨ctext⟩` is formatted.

21.2.6 Colors

For convenience, the `document-structure` 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

NotesSlides – 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 `notesslides` 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 `notesslides` 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 \LaTeX 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 `notesslides` 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 `notesslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

22.2.1 Package Options

The `notesslides` 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>	<ul style="list-style-type: none"> If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.
<code>showmeta</code>	<ul style="list-style-type: none"> <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 22.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>topsect</code>	<ul style="list-style-type: none"> <code>topsect=<sect></code> can be used to specify the top-level sectioning level; the default for <code><sect></code> is <code>section</code>.

22.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `notesslides` 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

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.

`notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

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

The default footer line of the `notesslides` 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 `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add \TeX notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.¹²

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

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

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

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


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

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

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

```
\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 \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. 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{sproblem}[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{sproblem}
\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).

```
Problem 0.1 (Fitting Elephants)
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{sproblem}[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{sproblem}

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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 \LaTeX source.

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

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

25.3 Messages and logging

```

43 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

25.4 Persistence

78 `<@@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

25.5 HTML Annotations

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

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

```

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

```

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

```

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

```



```

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

```

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

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

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

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

```

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

25.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

301 </package>

```

Chapter 26

STEX -MathHub Implementation

```
302 <*package>
303
304 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
305
306 <@@=stex_path>
307
308 Warnings and error messages
309 \msg_new:nnn{stex}{error/norepository}{
310   No~archive~#1~found~in~#2
311 }
312 \msg_new:nnn{stex}{error/notinarchive}{
313   Not~currently~in~an~archive,~but~\detokenize{#1}~
314   needs~one!
315 }
316 \msg_new:nnn{stex}{error/nofile}{
317   \detokenize{#1}~could~not~find~file~#2
318 }
319 \msg_new:nnn{stex}{error/twofiles}{
320   \detokenize{#1}~found~two~candidates~for~#2
321 }
```

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
320 \cs_new_protected:Nn \stex_path_from_string:Nn {
321   \str_set:Nx \l_tmpa_str { #2 }
322   \str_if_empty:NTF \l_tmpa_str {
323     \seq_clear:N #1
324   }{
325     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
326     \sys_if_platform_windows:T{
327       \seq_clear:N \l_tmpa_tl
```

```

328     \seq_map_inline:Nn #1 {
329       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
330       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
331     }
332     \seq_set_eq:NN #1 \l_tmpa_tl
333   }
334   \stex_path_canonicalize:N #1
335 }
336 }
337 \cs_generate_variant:Nn \stex_path_from_string:Nn
338 { NV, cn, cV }

```

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

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

```

339 \cs_new_protected:Nn \stex_path_to_string:NN {
340   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
341 }
342
343 \cs_new:Nn \stex_path_to_string:N {
344   \seq_use:Nn #1 /
345 }

```

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

`\c__stex_path_dot_str` . and .., respectively.
`\c__stex_path_up_str`

```

346 \str_const:Nn \c__stex_path_dot_str {.}
347 \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.

```

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

```

```

369         }{
370         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
371         }
372     }
373     }{
374     \str_if_empty:NF \l_tmpa_tl {
375     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
376     }
377     }
378 }
379 }
380 \seq_gset_eq:NN #1 \l_tmpa_seq
381 }
382 }

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

383 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
384   \seq_if_empty:NTF #1 {
385     \prg_return_false:
386   }{
387     \seq_get_left:NN #1 \l_tmpa_tl
388     \str_if_empty:NTF \l_tmpa_tl {
389       \prg_return_true:
390     }{
391       \prg_return_false:
392     }
393   }
394 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

395 \str_new:N\l_stex_kpsewhich_return_str
396 \cs_new_protected:Nn \stex_kpsewhich:n {
397   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
398   \exp_args:NNo \str_set:Nn \l_stex_kpsewhich_return_str{\l_tmpa_tl}
399   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
400 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

401 \sys_if_platform_windows:TF{
402   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
403 }{
404   \stex_kpsewhich:n{-var-value~PWD}
405 }
406

```



```

407 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
408 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
409 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

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

26.3 File Hooks and Tracking

```

410 <@=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 `STEX`-purposes.

`\g_stex_files_stack` keeps track of file changes

```

411 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

412 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
413 \stex_path_from_string:Nn \c_stex_mainfile_seq
414 \c_stex_mainfile_str

```

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

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

```

415 \seq_gclear_new:N\g_stex_currentfile_seq
416 \AddToHook{file/before}{
417   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
418   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
419     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
420   }{
421     \stex_path_from_string:Nn\g_stex_currentfile_seq{
422       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
423     }
424   }
425   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
426   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
427 }
428 \AddToHook{file/after}{
429   \seq_if_empty:NF\g_stex_files_stack{
430     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
431   }
432   \seq_if_empty:NTF\g_stex_files_stack{
433     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
434   }{
435     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
436     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
437   }
438 }

```

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

26.4 MathHub Repositories

```

439 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
440 \str_if_empty:NTF\mathhub{
441   \stex_kpsewhich:n{-var-value~MATHHUB}
442   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
443
444   \str_if_empty:NTF\c_stex_mathhub_str{
445     \msg_warning:nn{stex}{warning/nomathhub}
446   }{
447     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
448     \exp_args:NNx \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
449   }
450 }{
451   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
452   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
453     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
454       \c_stex_pwd_str/\mathhub
455     }
456   }
457   \stex_path_to_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
458   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
459 }

```

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

```

\__stex_mathhub_do_manifest:n
460 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
461   \str_set:Nx \l_tmpa_str { #1 }
462   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
463     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
464     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
465     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
466     \__stex_mathhub_find_manifest:N \l_tmpa_seq
467     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
468       \msg_error:nxxx{stex}{error/norepository}{#1}{
469         \stex_path_to_string:N \c_stex_mathhub_str
470       }
471     } {
472       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
473     }
474   }
475 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l_stex_mathhub_manifest_file_seq
476 \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:

```

477 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
478   \seq_set_eq:NN \l_tmpa_seq #1
479   \bool_set_true:N \l_tmpa_bool
480   \bool_while_do:Nn \l_tmpa_bool {
481     \seq_if_empty:NTF \l_tmpa_seq {
482       \bool_set_false:N \l_tmpa_bool
483     }{
484       \file_if_exist:nTF{
485         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
486       }{
487         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
488         \bool_set_false:N \l_tmpa_bool
489       }{
490         \file_if_exist:nTF{
491           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
492         }{
493           \seq_put_right:Nn \l_tmpa_seq{META-INF}
494           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
495           \bool_set_false:N \l_tmpa_bool
496         }{
497           \file_if_exist:nTF{
498             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
499           }{
500             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
501             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
502             \bool_set_false:N \l_tmpa_bool
503           }{
504             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
505           }
506         }
507       }
508     }
509   }
510   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
511 }

```

(End definition for _stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

512 \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:

```

513 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
514   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
515   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
516   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
517     \str_set:Nn \l_tmpa_str {##1}
518     \exp_args:NNoo \seq_set_split:Nnn

```

```

519     \l_tmpb_seq \c_colon_str \l_tmpa_str
520 \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
521   \exp_args:NNe \str_set:Nn \l_tmpb_tl {
522     \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
523   }
524   \exp_args:No \str_case:nnTF \l_tmpa_tl {
525     {id} {
526       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
527       { id } \l_tmpb_tl
528     }
529     {narration-base} {
530       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531       { narr } \l_tmpb_tl
532     }
533     {url-base} {
534       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535       { docurl } \l_tmpb_tl
536     }
537     {source-base} {
538       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539       { ns } \l_tmpb_tl
540     }
541     {ns} {
542       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543       { ns } \l_tmpb_tl
544     }
545     {dependencies} {
546       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547       { deps } \l_tmpb_tl
548     }
549   }{}{}
550 }{}
551 }
552 \ior_close:N \c__stex_mathhub_manifest_ior
553 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

554 \cs_new_protected:Nn \stex_set_current_repository:n {
555   \stex_require_repository:n { #1 }
556   \prop_set_eq:Nc \l_stex_current_repository_prop {
557     c_stex_mathhub_#1_manifest_prop
558   }
559 }

```

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

`\stex_require_repository:n`

```

560 \cs_new_protected:Nn \stex_require_repository:n {
561   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
562     \stex_debug:nn{mathhub}{Opening~archive:~#1}
563     \__stex_mathhub_do_manifest:n { #1 }
564     \exp_args:Nx \stex_add_to_sms:n {
565       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {

```

```

566         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
567         ns    = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
568         narr  = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
569         deps  = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
570     }
571 }
572 }
573 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

574 %\prop_new:N \l_stex_current_repository_prop
575
576 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
577 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
578   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
579 } {
580   \__stex_mathhub_parse_manifest:n { main }
581   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
582   \l_tmpa_str
583   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str _manifest_prop }
584   \c_stex_mathhub_main_manifest_prop
585   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
586   \stex_debug:nn{mathhub}{Current~repository:~
587     \prop_item:Nn \l_stex_current_repository_prop {id}
588   }
589 }

```

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

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

```

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

```

```

611     \prop_if_exist:NTF \l_stex_current_repository_prop {
612       \prop_item:Nn \l_stex_current_repository_prop { id }::~
613       \meaning\l_stex_current_repository_prop
614     }{
615       no~repository
616     }
617   }
618   \prop_if_exist:NTF \l_stex_current_repository_prop {
619     \stex_set_current_repository:n {
620       \prop_item:Nn \l_stex_current_repository_prop { id }
621     }
622   }{
623     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
624   }
625 }
626 }
627 }

```

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

```

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

```

```

661     \endgroup
662   }
663 }
664 }
665
666 \NewDocumentCommand \inputref { 0{} m}{
667   \stex_inputref:nn{ #1 }{ #2 }
668 }
669
670 \cs_new_protected:Nn \stex_mhbibresource:nn {
671   \stex_in_repository:nn {#1} {
672     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
673   }
674 }
675 \newcommand\addmhbibresource[2][]{
676   \stex_mhbibresource:nn{ #1 }{ #2 }
677 }

```

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

`\mhp`

```

678 \def \mhp #1 #2 {
679   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
680     \c_stex_mathhub_str /
681     \prop_item:Nn \l_stex_current_repository_prop { id }
682     / source / #2
683   }{
684     \c_stex_mathhub_str / #1 / source / #2
685   }
686 }

```

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

`\libinput`

```

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

```

```

707     }
708   }{}
709 }
710 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
711   / \l_tmpa_str / lib / #1.tex
712 }{
713   \bool_set_true:N \l_tmpa_bool
714   \tl_put_right:Nx \l_tmpa_tl {
715     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
716       / \l_tmpa_str / lib / #1.tex}
717   }
718 }{}
719 \bool_if:NF \l_tmpa_bool {
720   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
721 }
722 \l_tmpa_tl
723 }

```

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

`\libusepackage`

```

724 \NewDocumentCommand \libusepackage {0{ } m} {
725   \prop_if_exist:NF \l_stex_current_repository_prop {
726     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
727   }
728   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
729     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
730   }
731   \bool_set_false:N \l_libusepackage_bool
732   \tl_clear:N \l_tmpa_tl
733   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
734   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
735   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
736   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
737     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
738     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
739       / meta-inf / lib / #2.sty}{
740       \bool_set_true:N \l_libusepackage_bool
741       \tl_put_right:Nx \l_tmpa_tl {
742         \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
743           / meta-inf / lib / #2}
744       }
745     }{}
746   }
747   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
748     / \l_tmpa_str / lib / #2.sty
749   }{
750     \bool_if:NT \l_libusepackage_bool {
751       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
752     }
753     \bool_set_true:N \l_libusepackage_bool
754     \tl_put_right:Nx \l_tmpa_tl {
755       \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
756         / \l_tmpa_str / lib / #2}

```



```

757     }
758   }{}
759   \bool_if:NF \l_libusepackage_bool {
760     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
761   }
762   \l_tmpa_tl
763 }

```

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

```

764
765 \AddToHook{begindocument}{
766 \ltx@ifpackageloaded{graphicx}{
767   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
768   \newcommand\mhgraphics[2][]{\%
769     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
770     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
771   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
772 }{}
773 \ltx@ifpackageloaded{listings}{
774   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
775   \newcommand\lstinputmhlising[2][]{\%
776     \def\lst@mhrepos{}\setkeys{lst}{#1}%
777     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
778   \newcommand\clstinputmhlising[2][]{\begin{center}\lstinputmhlising[#1]{#2}\end{center}}
779 }{}
780 }
781
782
783 \</package>

```

Chapter 27

STEX -References Implementation

```
784 <*package>
785
786 %%%%%%%%%% references.dtx %%%%%%%%%%
787
788 %\RequirePackage{hyperref}
789 %\RequirePackage{cleveref}
790 <@@=stex_refs>
791
792 Warnings and error messages
793
794 \iow_new:N \c__stex_refs_refs_iow
795 \AddToHook{begindocument}{
796   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
797 }
798 \AddToHook{enddocument}{
799   \iow_close:N \c__stex_refs_refs_iow
800 }
801
802 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
803
804 \NewDocumentCommand \STEXreftitle { m } {
805   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
806 }
807
```

27.1 Document URIs and URLs

```
805 \seq_new:N \g__stex_refs_all_refs_seq
806
807 \str_new:N \l_stex_current_docns_str
808
809 \cs_new_protected:Nn \stex_get_document_uri: {
810   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
811   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
812   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
813   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
814 }
815
```

```

814 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
815
816 \str_clear:N \l_tmpa_str
817 \prop_if_exist:NT \l_stex_current_repository_prop {
818   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
819     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
820   }
821 }
822
823 \str_if_empty:NTF \l_tmpa_str {
824   \str_set:Nx \l_stex_current_docns_str {
825     file:/\stex_path_to_string:N \l_tmpa_seq
826   }
827 }{
828   \bool_set_true:N \l_tmpa_bool
829   \bool_while_do:Nn \l_tmpa_bool {
830     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
831     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
832       {source} { \bool_set_false:N \l_tmpa_bool }
833     }{}{
834       \seq_if_empty:NT \l_tmpa_seq {
835         \bool_set_false:N \l_tmpa_bool
836       }
837     }
838   }
839
840   \seq_if_empty:NTF \l_tmpa_seq {
841     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
842   }{
843     \str_set:Nx \l_stex_current_docns_str {
844       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
845     }
846   }
847 }
848 }
849
850 \str_new:N \l_stex_current_docurl_str
851 \cs_new_protected:Nn \stex_get_document_url: {
852   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
853   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
854   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
855   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
856   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
857
858   \str_clear:N \l_tmpa_str
859   \prop_if_exist:NT \l_stex_current_repository_prop {
860     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
861       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
862         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
863       }
864     }
865
866     \str_if_empty:NTF \l_tmpa_str {
867       \str_set:Nx \l_stex_current_docurl_str {

```

```

868     file:/\stex_path_to_string:N \l_tmpa_seq
869   }
870 }{
871   \bool_set_true:N \l_tmpa_bool
872   \bool_while_do:Nn \l_tmpa_bool {
873     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
874     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
875       {source} { \bool_set_false:N \l_tmpa_bool }
876     }{}{
877       \seq_if_empty:NT \l_tmpa_seq {
878         \bool_set_false:N \l_tmpa_bool
879       }
880     }
881   }
882
883   \seq_if_empty:NTF \l_tmpa_seq {
884     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
885   }{
886     \str_set:Nx \l_stex_current_docurl_str {
887       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
888     }
889   }
890 }
891 }

```

27.2 Setting Reference Targets

```

892 \str_const:Nn \c__stex_refs_url_str{URL}
893 \str_const:Nn \c__stex_refs_ref_str{REF}
894 % @currentlabel -> number
895 % @currentlabelname -> title
896 % @currentHref -> name.number <- id of some kind
897 % \theH# -> \arabic{section}
898 % \the# -> number
899 % \hyper@makecurrent{#}
900 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
901   \stex_get_document_uri:
902   \str_set:Nx \l_tmpa_str { #1 }
903   \str_if_empty:NT \l_tmpa_str {
904     \int_zero:N \l_tmpa_int
905     \bool_set_true:N \l_tmpa_bool
906     \bool_while_do:Nn \l_tmpa_bool {
907       \cs_if_exist:cTF {
908         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
909       }{
910         \int_incr:N \l_tmpa_int
911       }{
912         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
913         \bool_set_false:N \l_tmpa_bool
914       }
915     }
916   }
917   \str_set:Nx \l_tmpa_str {
918     \l_stex_current_docns_str??\l_tmpa_str

```

```

919 }
920 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
921 \stex_if_smsmode:TF {
922   \stex_get_document_url:
923   \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
924   \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
925 }{
926   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
927   \exp_args:Nx\label{sref_\l_tmpa_str}
928   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
929   \str_gset:cx {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
930 }
931 }
932 \cs_new_protected:Npn \stexauxadddocref #1 {
933   \str_set:Nx \l_tmpa_str {#1}
934   \str_gset_eq:cN{sref_\l_tmpa_str_type}\c__stex_refs_ref_str
935   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
936 }
937 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
938   \stex_get_document_uri:
939   \stex_if_smsmode:TF {
940     \stex_get_document_url:
941     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
942     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
943   }
944   {
945     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
946     \exp_args:Nx\label{sref_sym_#1}
947   }
948   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
949   \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
950 }
951 }

```

27.3 Using References

```

952 \str_new:N \l__stex_refs_indocument_str
953 \keys_define:nn { stex / sref } {
954   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
955   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
956   pre           .tl_set:N = \l__stex_refs_pre_tl ,
957   post          .tl_set:N = \l__stex_refs_post_tl ,
958   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
959 }
960
961 \bool_new:N \c__stex_refs_hyperref_bool
962 \bool_set_false:N \c__stex_refs_hyperref_bool
963 \AddToHook{begindocument}{
964   \@ifpackageloaded{hyperref}{
965     \bool_set_true:N \c__stex_refs_hyperref_bool
966   }{}
967 }
968
969

```

```

970 \cs_new_protected:Nn \__stex_refs_args:n {
971   \tl_clear:N \l__stex_refs_linktext_tl
972   \tl_clear:N \l__stex_refs_fallback_tl
973   \tl_clear:N \l__stex_refs_pre_tl
974   \tl_clear:N \l__stex_refs_post_tl
975   \str_clear:N \l__stex_refs_repo_str
976   \keys_set:nn { stex / sref } { #1 }
977 }
978
979 \NewDocumentCommand \sref { 0{} m}{
980   \__stex_refs_args:n { #1 }
981   \str_if_empty:NTF \l__stex_refs_indocument_str {
982     \str_set:Nn \l_tmpa_str { #2 }
983     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
984     \tl_set:Nn \l_tmpa_tl {
985       \l__stex_refs_fallback_tl
986     }
987     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
988       \str_set:Nn \l_tmpb_str { ##1 }
989       \str_if_eq:eeT { \l_tmpa_str } {
990         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
991       } {
992         \seq_map_break:n {
993           \tl_set:Nn \l_tmpa_tl {
994             % doc uri in \l_tmpb_str
995             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
996             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
997               % reference
998               \cs_if_exist:cTF{autoref}{
999                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1000               }{
1001                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1002               }
1003             }{
1004               % URL
1005               \if_bool:N \c__stex_refs_hyperref_bool {
1006                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1007               }{
1008                 \l__stex_refs_fallback_tl
1009               }
1010             }
1011           }
1012         }
1013       }
1014     }
1015     \l_tmpa_tl
1016   }{
1017     % TODO
1018   }
1019 }
1020
1021 \NewDocumentCommand \srefsym { 0{} m}{
1022   \stex_get_symbol:n { #2 }
1023   \__stex_refs_args:n { #1 }

```

```

1024 \str_if_empty:NTF \l__stex_refs_indocument_str {
1025   \tl_set:Nn \l_tmpa_tl {
1026     \l__stex_refs_fallback_tl
1027   }
1028   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1029     \tl_set:Nn \l_tmpa_tl {
1030       % doc uri in \l_tmpb_str
1031       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1032       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1033         % reference
1034         \cs_if_exist:cTF{autoref}{
1035           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1036         }{
1037           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1038         }
1039       }{
1040         % URL
1041         \if_bool:N \c__stex_refs_hyperref_bool {
1042           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1043         }{
1044           \l__stex_refs_fallback_tl
1045         }
1046       }
1047     }
1048   }
1049   \l_tmpa_tl
1050 }{
1051   % TODO
1052 }
1053 }
1054
1055 \cs_new_protected:Npn \srefsymuri #1 #2 {
1056   \hyperref[sref_sym_#1]{#2}
1057 }
1058
1059 </package>

```

Chapter 28

STEX -Modules Implementation

```
1060 <*package>
1061
1062 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1063
1064 <@@=stex_modules>
1065
1066   Warnings and error messages
1067   \msg_new:nnn{stex}{error/unknownmodule}{
1068     No~module~#1~found
1069   }
1070   \msg_new:nnn{stex}{error/syntax}{
1071     Syntax~error:~#1
1072   }
1073   \msg_new:nnn{stex}{error/siglanguage}{
1074     Module~#1~declares~signature~#2,~but~does~not~
1075     declare~its~language
1076   }
1077   \msg_new:nnn{stex}{error/concllictingmodules}{
1078     Comflicting~imports~for~module~#1
1079   }
1080
1081 \l_stex_current_module_str The current module:
1082 \str_new:N \l_stex_current_module_str
1083
1084 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1085
1086 \l_stex_all_modules_seq Stores all available modules
1087 \seq_new:N \l_stex_all_modules_seq
1088
1089 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1090
1091 \stex_if_in_module_p:
1092 \stex_if_in_module:TF
1093 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1094   \str_if_empty:NTF \l_stex_current_module_str
1095     \prg_return_false: \prg_return_true:
1096 }
```


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

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

```
1085 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1086   \prop_if_exist:cTF { c_stex_module_#1_prop }
1087   \prg_return_true: \prg_return_false:
1088 }
```

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1089 \cs_new_protected:Nn \stex_add_to_current_module:n {
1090   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1091 }
1092 \cs_new_protected:Npn \STEXexport {
1093   \begingroup
1094   \newlinechar=-1\relax
1095   \endlinechar=-1\relax
1096   %\catcode'\ = 9\relax
1097   \expandafter\endgroup\STEXexport:n
1098 }
1099 \cs_new_protected:Nn \STEXexport:n {
1100   \ignorespaces #1
1101   \stex_add_to_current_module:n { \ignorespaces #1 }
1102   \stex_smsmode_set_codes:
1103 }
1104 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

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

`\stex_add_constant_to_current_module:n`

```
1105 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1106   \str_set:Nx \l_tmpa_str { #1 }
1107   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1108 }
1109
1110 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1111 % \str_set:Nx \l_tmpa_str { #1 }
1112 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1113 %}
```

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

`\stex_collect_imports:n`

```
1114 \cs_new_protected:Nn \stex_collect_imports:n {
1115   \seq_clear:N \l_stex_collect_imports_seq
1116   \__stex_modules_collect_imports:n {#1}
1117 }
1118 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1119   \seq_map_inline:cn {c_stex_module_#1_imports} {
1120     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1121       \__stex_modules_collect_imports:n { ##1 }
1122     }
1123 }
```

```

1123 }
1124 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1125   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1126 }
1127 }

```

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

`\stex_add_import_to_current_module:n`

```

1128 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1129   \str_set:Nx \l_tmpa_str { #1 }
1130   \exp_args:Nno
1131   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1132     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1133   }
1134 }

```

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

`\stex_modules_compute_namespace:nN`

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

```

1135 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1136   \str_set:Nx \l_tmpa_str { #1 }
1137   \seq_set_eq:NN \l_tmpa_seq #2
1138   % split off file extension
1139   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1140   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1141   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1142   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1143
1144   \bool_set_true:N \l_tmpa_bool
1145   \bool_while_do:Nn \l_tmpa_bool {
1146     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1147     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1148       {source} { \bool_set_false:N \l_tmpa_bool }
1149     }{}{
1150       \seq_if_empty:NT \l_tmpa_seq {
1151         \bool_set_false:N \l_tmpa_bool
1152       }
1153     }
1154   }
1155
1156   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1157   \str_if_empty:NTF \l_stex_modules_subpath_str {
1158     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1159   }{
1160     \str_set:Nx \l_stex_modules_ns_str {
1161       \l_tmpa_str/\l_stex_modules_subpath_str
1162     }
1163   }
1164 }

```

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

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1165 \str_new:N \l_stex_modules_ns_str
1166 \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.

```

1167 \cs_new_protected:Nn \stex_modules_current_namespace: {
1168   \str_clear:N \l_stex_modules_subpath_str
1169   \prop_if_exist:NTF \l_stex_current_repository_prop {
1170     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1171     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1172   }{
1173     % split off file extension
1174     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1175     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1176     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1177     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1178     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1179     \str_set:Nx \l_stex_modules_ns_str {
1180       file:/\stex_path_to_string:N \l_tmpa_seq
1181     }
1182   }
1183 }

```

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

28.1 The module environment

module arguments:

```

1184 \keys_define:nn { stex / module } {
1185   title      .str_set_x:N = \l_stex_module_title_str ,
1186   ns         .str_set_x:N = \l_stex_module_ns_str ,
1187   lang       .str_set_x:N = \l_stex_module_lang_str ,
1188   sig        .str_set_x:N = \l_stex_module_sig_str ,
1189   creators   .str_set_x:N = \l_stex_module_creators_str ,
1190   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1191   meta       .str_set_x:N = \l_stex_module_meta_str ,
1192   srccite    .str_set_x:N = \l_stex_module_srccite_str
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_args:n {
1196   \str_clear:N \l_stex_module_title_str
1197   \str_clear:N \l_stex_module_ns_str
1198   \str_clear:N \l_stex_module_lang_str
1199   \str_clear:N \l_stex_module_sig_str
1200   \str_clear:N \l_stex_module_creators_str
1201   \str_clear:N \l_stex_module_contributors_str
1202   \str_clear:N \l_stex_module_meta_str
1203   \str_clear:N \l_stex_module_srccite_str
1204   \keys_set:nn { stex / module } { #1 }

```

```

1205 }
1206
1207 % module parameters here? In the body?
1208

```

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

```

1209 \cs_new_protected:Nn \stex_module_setup:nn {
1210   \str_set:Nx \l_stex_module_name_str { #2 }
1211   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1212 \stex_if_in_module:TF {
1213   % Nested module
1214   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1215   { ns } \l_stex_module_ns_str
1216   \str_set:Nx \l_stex_module_name_str {
1217     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1218     { name } / \l_stex_module_name_str
1219   }
1220 }{
1221   % not nested:
1222   \str_if_empty:NT \l_stex_module_ns_str {
1223     \stex_modules_current_namespace:
1224     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1225     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1226     / {\l_stex_module_ns_str}
1227     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1228     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1229       \str_set:Nx \l_stex_module_ns_str {
1230         \stex_path_to_string:N \l_tmpa_seq
1231       }
1232     }
1233   }
1234 }

```

Next, we determine the language of the module:

```

1235 \str_if_empty:NT \l_stex_module_lang_str {
1236   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1237   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1238   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1239   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1240   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1241     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1242       inferred~from~file~name}
1243     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1244   }
1245 }
1246
1247 \str_if_empty:NF \l_stex_module_lang_str {
1248   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1249   \l_tmpa_str {
1250     \ltx@ifpackageloaded{babel}{
1251       \exp_args:Nx \selectlanguage { \l_tmpa_str }

```

```

1252     }{}
1253   } {
1254     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1255   }
1256 }

```

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

```

1257 \str_if_empty:NTF \l_stex_module_sig_str {
1258   \exp_args:Nnx \prop_gset_from_keyval:cn {
1259     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1260   } {
1261     name      = \l_stex_module_name_str ,
1262     ns        = \l_stex_module_ns_str ,
1263     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1264     lang      = \l_stex_module_lang_str ,
1265     sig       = \l_stex_module_sig_str ,
1266     meta      = \l_stex_module_meta_str
1267   }
1268   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1269   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1270   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1271   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1272   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1273 \str_if_empty:NT \l_stex_module_meta_str {
1274   \str_set:Nx \l_stex_module_meta_str {
1275     \c_stex_metatheory_ns_str ? Metatheory
1276   }
1277 }
1278 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1279   \bool_set_true:N \l_stex_in_meta_bool
1280   \exp_args:Nx \stex_add_to_current_module:n {
1281     \bool_set_true:N \l_stex_in_meta_bool
1282     \stex_activate_module:n {\l_stex_module_meta_str}
1283     \bool_set_false:N \l_stex_in_meta_bool
1284   }
1285   \stex_activate_module:n {\l_stex_module_meta_str}
1286   \bool_set_false:N \l_stex_in_meta_bool
1287 }
1288 }{
1289   \str_if_empty:NT \l_stex_module_lang_str {
1290     \msg_error:nnxx{stex}{error/siglanguage}{
1291       \l_stex_module_ns_str?\l_stex_module_name_str
1292     }{\l_stex_module_sig_str}
1293   }
1294
1295   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1296   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1297   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1298   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1299   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1300   \str_set:Nx \l_tmpa_str {

```

```

1301     \stex_path_to_string:N \l_tmpa_seq /
1302     \l_tmpa_str . \l_stex_module_sig_str .tex
1303   }
1304   \IfFileExists \l_tmpa_str {
1305     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1306       \seq_clear:N \l_stex_all_modules_seq
1307       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1308       \input { \l_tmpa_str }
1309     }
1310   }{
1311     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1312   }
1313   \stex_if_smsmode:F {
1314     \stex_activate_module:n {
1315       \l_stex_module_ns_str ? \l_stex_module_name_str
1316     }
1317   }
1318   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1319 }
1320 }

```

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

module The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1321 \int_new:N \l_stex_module_group_depth_int
1322 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1323   \stex_reactivate_macro:N \STEXexport
1324   \stex_reactivate_macro:N \importmodule
1325   \stex_reactivate_macro:N \symdecl
1326   \stex_reactivate_macro:N \notation
1327   \stex_reactivate_macro:N \symdef
1328   \stex_module_setup:nn{#1}{#2}
1329 }
1330 \stex_debug:nn{modules}{
1331   New~module:\\
1332   Namespace:~\l_stex_module_ns_str\\
1333   Name:~\l_stex_module_name_str\\
1334   Language:~\l_stex_module_lang_str\\
1335   Signature:~\l_stex_module_sig_str\\
1336   Metatheory:~\l_stex_module_meta_str\\
1337   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1338 }
1339 }
1340 \seq_put_right:Nx \l_stex_all_modules_seq {
1341   \l_stex_module_ns_str ? \l_stex_module_name_str
1342 }
1343 }
1344 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1345 % { \l_stex_module_ns_str ? \l_stex_module_name_str }
1346 }
1347 }
1348 \stex_if_smsmode:TF {

```

```

1349 \stex_smsmode_set_codes:
1350 } {
1351 \begin{stex_annotate_env} {theory} {
1352 \l_stex_module_ns_str ? \l_stex_module_name_str
1353 }
1354
1355 \stex_annotate_invisible:nnn{header}{} {
1356 \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1357 \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1358 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1359 \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1360 }
1361 }
1362 }
1363 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1364 % TODO: Inherit metatheory for nested modules?
1365 }
1366 \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}

```

1367 \cs_new_protected:Nn \_stex_modules_end_module: {
1368 % \str_set:Nx \l_tmpa_str {
1369 % c_stex_module_
1370 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1371 % \prop_item:Nn \l_stex_current_module_prop { name }
1372 % _prop
1373 % }
1374 %^^A \prop_new:c { \l_tmpa_str }
1375 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1376 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1377 }

```

(End definition for _stex_modules_end_module:.)

@module The core environment, with no header

```

1378 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1379 \NewDocumentEnvironment { @module } { 0 } { m } {
1380 \par
1381 \_stex_modules_begin_module:nn{#1}{#2}
1382 } {
1383 \_stex_modules_end_module:
1384 \stex_if_smsmode:TF {
1385 % \exp_args:Nx \stex_add_to_sms:n {
1386 % \prop_gset_from_keyval:cn {
1387 % c_stex_module_
1388 % \prop_item:Nn \l_stex_current_module_prop { ns } ?
1389 % \prop_item:Nn \l_stex_current_module_prop { name }
1390 % _prop
1391 % } {
1392 % name = \prop_item:cn { \l_tmpa_str } { name } ,
1393 % ns = \prop_item:cn { \l_tmpa_str } { ns } ,
1394 % file = \prop_item:cn { \l_tmpa_str } { file } ,

```

```

1395 %      lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1396 %      sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1397 %      meta      = \prop_item:cn { \l_tmpa_str } { meta }
1398 %    }
1399 %  }
1400 }{
1401   \end{stex_annotate_env}
1402 }
1403 }

```

\stex_modules_heading: Code for document headers

```

1404 \cs_if_exist:NTF \thesection {
1405   \newcounter{module}[section]
1406 }{
1407   \newcounter{module}
1408 }
1409
1410 \bool_if:NT \c_stex_showmods_bool {
1411   \latexml_if:F { \RequirePackage{mdframed} }
1412 }
1413
1414 \cs_new_protected:Nn \stex_modules_heading: {
1415   \stepcounter{module}
1416   \par
1417   \bool_if:NT \c_stex_showmods_bool {
1418     \noindent{\textbf{Module} ~
1419       \cs_if_exist:NT \thesection {\thesection.}
1420       \themodule ~ [\l_stex_module_name_str]
1421     }
1422     \str_if_empty:NTF \l_stex_module_title_str {
1423       }{
1424         \quad(\l_stex_module_title_str)\hfill
1425       }\par
1426     }
1427     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1428     % TODO
1429     \stex_ref_new_doc_target:n \l_stex_module_name_str
1430   }

```

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

Finally:

```

1431 \NewDocumentEnvironment { module } { 0 } { m } {
1432   \bool_if:NT \c_stex_showmods_bool {
1433     \begin{mdframed}
1434   }
1435   \begin{@module}[#1]{#2}
1436     \stex_modules_heading:
1437   }{
1438     \end{@module}
1439     \bool_if:NT \c_stex_showmods_bool {
1440       \end{mdframed}
1441     }
1442   }

```


28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1443 \NewDocumentCommand \STEXModule { m } {
1444   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1445   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1446   \tl_set:Nn \l_tmpa_tl {
1447     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1448   }
1449   \seq_map_inline:Nn \l_stex_all_modules_seq {
1450     \str_set:Nn \l_tmpb_str { ##1 }
1451     \str_if_eq:eeT { \l_tmpa_str } {
1452       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1453     } {
1454       \seq_map_break:n {
1455         \tl_set:Nn \l_tmpa_tl {
1456           \stex_invoke_module:n { ##1 }
1457         }
1458       }
1459     }
1460   }
1461   \l_tmpa_tl
1462 }
1463
1464 \cs_new_protected:Nn \stex_invoke_module:n {
1465   \stex_debug:nn{modules}{Invoking~module~#1}
1466   \peek_charcode_remove:NTF ! {
1467     \__stex_modules_invoke_uri:nN { #1 }
1468   } {
1469     \peek_charcode_remove:NTF ? {
1470       \__stex_modules_invoke_symbol:nn { #1 }
1471     } {
1472       \msg_error:nnx{stex}{error/syntax}{
1473         ?~or~!~expected~after~
1474         \c_backslash_str STEXModule{#1}
1475       }
1476     }
1477   }
1478 }
1479
1480 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1481   \str_set:Nn #2 { #1 }
1482 }
1483
1484 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1485   \stex_invoke_symbol:n{#1?#2}
1486 }

```

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

```

\stex_activate_module:n
1487 \bool_new:N \l_stex_in_meta_bool
1488 \bool_set_false:N \l_stex_in_meta_bool

```

```

1489 \cs_new_protected:Nn \stex_activate_module:n {
1490   \stex_debug:nn{modules}{Activating~module~#1}
1491   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1492     \msg_error:nnn{stex}{error/concllictingmodules}{ #1 }
1493   }
1494   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1495     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1496     \use:c{ c_stex_module_#1_code }
1497   }
1498 }

```

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

```

1499 </package>

```

Chapter 29

STEX -Module Inheritance Implementation

```
1500 <*package>
1501
1502 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1503
```

29.1 SMS Mode

```
1504 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1505 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1506 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1507 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1508
1509 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1510   \makeatletter
1511   \makeatother
1512   \ExplSyntaxOn
1513   \ExplSyntaxOff
1514 }
1515
1516 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1517   \symdef
1518   \importmodule
1519   \notation
1520   \symdecl
1521   \STEXexport
1522 }
1523
1524 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1525   \tl_to_str:n {
1526     module,
1527     @module
```

```

1528 }
1529 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 31.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
1530 \bool_new:N \g__stex_smsmode_bool
1531 \bool_set_false:N \g__stex_smsmode_bool
1532 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1533   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1534 }

```

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

```

\__stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\__stex_smsmode_if_catcodes:TF
1535 \bool_new:N \g__stex_smsmode_catcode_bool
1536 \bool_set_false:N \g__stex_smsmode_catcode_bool
1537 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1538   \bool_if:NTF \g__stex_smsmode_catcode_bool
1539   \prg_return_true: \prg_return_false:
1540 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:
1541 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1542   \stex_if_smsmode:T {
1543     \__stex_smsmode_if_catcodes:F {
1544       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1545       \exp_after:wN \char_gset_active_eq:NN
1546       \c_backslash_str \__stex_smsmode_cs:
1547       \tex_global:D \char_set_catcode_active:N \
1548       \tex_global:D \char_set_catcode_other:N $
1549       \tex_global:D \char_set_catcode_other:N ^
1550       \tex_global:D \char_set_catcode_other:N _
1551       \tex_global:D \char_set_catcode_other:N &
1552       \tex_global:D \char_set_catcode_other:N ##
1553     }
1554   }
1555 } \iffalse $ \fi % to make syntax highlighting work again

```

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

```

\__stex_smsmode_unset_codes: Sets category code scheme back from the one used in SMS mode.
1556 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1557   \__stex_smsmode_if_catcodes:T {
1558     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1559     \exp_after:wN \tex_global:D \exp_after:wN
1560     \char_set_catcode_escape:N \c_backslash_str
1561     \tex_global:D \char_set_catcode_math_toggle:N $
1562     \tex_global:D \char_set_catcode_math_superscript:N ^
1563     \tex_global:D \char_set_catcode_math_subscript:N _
1564     \tex_global:D \char_set_catcode_alignment:N &
1565     \tex_global:D \char_set_catcode_parameter:N ##
1566   }
1567 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1568 \cs_new_protected:Nn \stex_in_smsmode:nn {
1569   \vbox_set:Nn \l_tmpa_box {
1570     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1571     \bool_gset_true:N \g__stex_smsmode_bool
1572     \stex_smsmode_set_codes:
1573     #2
1574     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1575     \stex_if_smsmode:F {
1576       \__stex_smsmode_unset_codes:
1577     }
1578   }
1579   \box_clear:N \l_tmpa_box
1580 }

```

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

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

```

1581 \cs_new_protected:Nn \_stex_smsmode_cs: {
1582   \str_clear:N \l_tmpa_str
1583   \peek_analysis_map_inline:n {
1584     % #1: token (one expansion)
1585     % #2: charcode
1586     % #3 catcode
1587     \token_if_eq_charcode:NNTF ##3 B {
1588       % token is a letter
1589       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1590     } {
1591       \str_if_empty:NTF \l_tmpa_str {
1592         % we don't allow (or need) single non-letter CSs
1593         % for now
1594         \peek_analysis_map_break:
1595       }{
1596         \str_if_eq:onTF \l_tmpa_str { begin } {
1597           \peek_analysis_map_break:n {
1598             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1599           }
1600         } {
1601           \str_if_eq:onTF \l_tmpa_str { end } {
1602             \peek_analysis_map_break:n {
1603               \exp_after:wN \_stex_smsmode_checkend:n ##1
1604             }
1605           } {
1606             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1607             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1608               \g_stex_smsmode_allowedmacros_tl
1609               { \use:c{\l_tmpa_str} } {
1610               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1611               \peek_analysis_map_break:n {
1612                 \exp_after:wN \l_tmpa_tl ##1
1613               }

```

```

1614     } {
1615         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1616         \g_stex_smsmode_allowedmacros_escape_tl
1617         { \use:c{\l_tmpa_str} } {
1618             \__stex_smsmode_unset_codes:
1619             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1620             % TODO \__stex_smsmode_rescan_cs:
1621             \int_compare:nNnTF {##2} = {92} {
1622                 \peek_analysis_map_break:n {
1623                     \__stex_smsmode_unset_codes:
1624                     \__stex_smsmode_rescan_cs:
1625                 }
1626             } {
1627                 \peek_analysis_map_break:n {
1628                     \exp_after:wN \l_tmpa_tl ##1
1629                 }
1630             }
1631         } {
1632             \int_compare:nNnTF {##2} = {92} {
1633                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1634             } {
1635                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1636             }
1637         }
1638     }
1639 }
1640 }
1641 }
1642 }
1643 }
1644 }

```

(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:

```

1645 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1646     \str_clear:N \l_tmpb_str
1647     \peek_analysis_map_inline:n {
1648         \token_if_eq_charcode:NNTF ##3 B {
1649             % token is a letter
1650             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1651         } {
1652             \peek_analysis_map_break:n {
1653                 \exp_after:wN \use:c \exp_after:wN {
1654                     \exp_after:wN \l_tmpa_str\exp_after:wN
1655                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1656             }
1657         }
1658     }
1659 }

```

(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.

```

1660 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1661   \str_set:Nn \l_tmpa_str { #1 }
1662   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1663     \__stex_smsmode_unset_codes:
1664     \begin{#1}
1665   }
1666 }
```

(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.

```

1667 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1668   \str_set:Nn \l_tmpa_str { #1 }
1669   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1670     \end{#1}
1671   }
1672 }
```

(End definition for `__stex_smsmode_checkend:n`.)

29.2 Inheritance

1673 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1674 \cs_new_protected:Nn \stex_import_module_uri:nn {
1675   \str_set:Nx \l_stex_import_archive_str { #1 }
1676   \str_set:Nn \l_stex_import_path_str { #2 }
1677
1678   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1679   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1680   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1681
1682   \stex_modules_current_namespace:
1683   \bool_lazy_all:nTF {
1684     {\str_if_empty_p:N \l_stex_import_archive_str}
1685     {\str_if_empty_p:N \l_stex_import_path_str}
1686     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1687   }{
1688     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1689     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1690   }{
1691     \str_if_empty:NT \l_stex_import_archive_str {
1692       \prop_if_exist:NT \l_stex_current_repository_prop {
1693         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1694       }
1695     }
1696     \str_if_empty:NTF \l_stex_import_archive_str {
1697       \str_if_empty:NF \l_stex_import_path_str {
1698         \str_set:Nx \l_stex_import_ns_str {
1699           \l_stex_module_ns_str / \l_stex_import_path_str
1700         }
1701       }
1702     }
```

```

1702   }{
1703     \stex_require_repository:n \l_stex_import_archive_str
1704     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str _manifest_prop } { ns }
1705     \l_stex_import_ns_str
1706     \str_if_empty:NF \l_stex_import_path_str {
1707       \str_set:Nx \l_stex_import_ns_str {
1708         \l_stex_import_ns_str / \l_stex_import_path_str
1709       }
1710     }
1711   }
1712 }
1713 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1714 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1715 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1716 \str_new:N \l_stex_import_path_str
                            1717 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1718 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1719   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1720
1721     % archive
1722     \str_set:Nx \l_tmpa_str { #2 }
1723     \str_if_empty:NTF \l_tmpa_str {
1724       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1725     } {
1726       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1727       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1728       \seq_put_right:Nn \l_tmpa_seq { source }
1729     }
1730
1731     % path
1732     \str_set:Nx \l_tmpb_str { #3 }
1733     \str_if_empty:NTF \l_tmpb_str {
1734       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1735
1736       \ltx@ifpackageloaded{babel} {
1737         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1738           { \language } \l_tmpb_str {
1739           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1740         }
1741       } {
1742         \str_clear:N \l_tmpb_str
1743       }
1744
1745       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1746       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1747         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1748 }{
1749   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1750   \IfFileExists{ \l_tmpa_str.tex }{
1751     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1752   }{
1753     % try english as default
1754     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1755     \IfFileExists{ \l_tmpa_str.en.tex }{
1756       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1757     }{
1758       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1759     }
1760   }
1761 }
1762
1763 } {
1764   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1765   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1766
1767   \ltx@ifpackageloaded{babel} {
1768     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1769       { \language } \l_tmpb_str {
1770       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1771     }
1772   } {
1773     \str_clear:N \l_tmpb_str
1774   }
1775
1776   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1777
1778   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1779   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1780     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1781   }{
1782     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1783     \IfFileExists{ \l_tmpa_str/#4.tex }{
1784       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1785     }{
1786       % try english as default
1787       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1788       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1789         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1790       }{
1791         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1792         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1793           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1794         }{
1795           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1796           \IfFileExists{ \l_tmpa_str.tex }{
1797             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1798           }{
1799             % try english as default
1800             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1801             \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1802         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1803     }{
1804         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1805     }
1806 }
1807 }
1808 }
1809 }
1810 }
1811 }
1812
1813 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1814     \seq_clear:N \l_stex_all_modules_seq
1815     \str_clear:N \l_stex_current_module_str
1816     \str_set:Nx \l_tmpb_str { #2 }
1817     \str_if_empty:NF \l_tmpb_str {
1818         \stex_set_current_repository:n { #2 }
1819     }
1820     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1821     \input { \g__stex_importmodule_file_str }
1822 }
1823
1824 \stex_if_module_exists:nF { #1 ? #4 } {
1825     \msg_error:nnx{stex}{error/unknownmodule}{
1826         #1?#4~(in~file~\g__stex_importmodule_file_str)
1827     }
1828 }
1829 }
1830 \stex_activate_module:n { #1 ? #4 }
1831 }

```

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

`\importmodule`

```

1832 \NewDocumentCommand \importmodule { 0{} m } {
1833     \stex_import_module_uri:nn { #1 } { #2 }
1834     \stex_debug:nn{modules}{Importing~module:~
1835         \l_stex_import_ns_str ? \l_stex_import_name_str
1836     }
1837     \stex_if_smsmode:F {
1838         \stex_import_require_module:nnnn
1839         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1840         { \l_stex_import_path_str } { \l_stex_import_name_str }
1841         \stex_annotate_invisible:nnn
1842         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1843     }
1844     \exp_args:Nx \stex_add_to_current_module:n {
1845         \stex_import_require_module:nnnn
1846         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1847         { \l_stex_import_path_str } { \l_stex_import_name_str }
1848     }
1849     \exp_args:Nx \stex_add_import_to_current_module:n {
1850         \l_stex_import_ns_str ? \l_stex_import_name_str
1851     }

```

```

1852 \stex_smsmode_set_codes:
1853 }
1854 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1855 \NewDocumentCommand \usemodule { 0{} m } {
1856   \stex_if_smsmode:F {
1857     \stex_import_module_uri:nn { #1 } { #2 }
1858     \stex_import_require_module:nnnn
1859     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1860     { \l_stex_import_path_str } { \l_stex_import_name_str }
1861     \stex_annotate_invisible:nnn
1862     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1863   }
1864   \stex_smsmode_set_codes:
1865 }

```

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

```

1866 \endpackage

```

Chapter 30

STEX -Symbols Implementation

```
1867 <*package>
1868
1869 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1870
```

Warnings and error messages

```
1871
```

30.1 Symbol Declarations

```
1872 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1873 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1874 \NewDocumentCommand \STEXsymbol { m } {
1875   \stex_get_symbol:n { #1 }
1876   \exp_args:No
1877   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1878 }
```

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

symdecl arguments:

```
1879 \keys_define:nn { stex / symdecl } {
1880   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1881   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1882   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1883   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1884   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1885   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1886   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1887   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1888 }
```

```

1889
1890 \bool_new:N \l_stex_symdecl_make_macro_bool
1891
1892 \cs_new_protected:Nn \__stex_symdecl_args:n {
1893   \str_clear:N \l_stex_symdecl_name_str
1894   \str_clear:N \l_stex_symdecl_args_str
1895   \bool_set_false:N \l_stex_symdecl_local_bool
1896   \tl_clear:N \l_stex_symdecl_type_tl
1897   \tl_clear:N \l_stex_symdecl_definiens_tl
1898
1899   \keys_set:nn { stex / symdecl } { #1 }
1900 }

```

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

```

1901
1902 \NewDocumentCommand \symdecl { s O{} m } {
1903   \__stex_symdecl_args:n { #2 }
1904   \IfBooleanTF #1 {
1905     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1906   } {
1907     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1908   }
1909   \stex_symdecl_do:n { #3 }
1910   \stex_smsmode_set_codes:
1911 }
1912 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1913 \cs_new_protected:Nn \stex_symdecl_do:n {
1914   \stex_if_in_module:F {
1915     % TODO throw error? some default namespace?
1916   }
1917
1918   \str_if_empty:NT \l_stex_symdecl_name_str {
1919     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1920   }
1921
1922   \prop_if_exist:cT { l_stex_symdecl_
1923     \l_stex_current_module_str ?
1924     \l_stex_symdecl_name_str
1925     _prop
1926   }{
1927     % TODO throw error (beware of circular dependencies)
1928   }
1929
1930   \prop_clear:N \l_tmpa_prop
1931   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1932   \seq_clear:N \l_tmpa_seq
1933   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1934   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1935

```

```

1936 \exp_args:No \stex_add_constant_to_current_module:n {
1937   \l_stex_symdecl_name_str
1938 }
1939
1940 % arity/args
1941 \int_zero:N \l_tmpb_int
1942
1943 \bool_set_true:N \l_tmpa_bool
1944 \str_map_inline:Nn \l_stex_symdecl_args_str {
1945   \token_case_meaning:NnF ##1 {
1946     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1947     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1948     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1949     {\tl_to_str:n a} {
1950       \bool_set_false:N \l_tmpa_bool
1951       \int_incr:N \l_tmpb_int
1952     }
1953     {\tl_to_str:n B} {
1954       \bool_set_false:N \l_tmpa_bool
1955       \int_incr:N \l_tmpb_int
1956     }
1957   }{
1958     \msg_set:nnn{stex}{error/wrongargs}{
1959       args~value~in~symbol~declaration~for~
1960       \l_stex_current_module_str ?
1961       \l_stex_symdecl_name_str ~
1962       needs~to~be~
1963       i,~a,~b~or~B,~but~##1~given
1964     }
1965     \msg_error:nn{stex}{error/wrongargs}
1966   }
1967 }
1968 \bool_if:NTF \l_tmpa_bool {
1969   % possibly numeric
1970   \str_if_empty:NTF \l_stex_symdecl_args_str {
1971     \prop_put:Nnn \l_tmpa_prop { args } {}
1972     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1973   }{
1974     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1975     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1976     \str_clear:N \l_tmpa_str
1977     \int_step_inline:nn \l_tmpa_int {
1978       \str_put_right:Nn \l_tmpa_str i
1979     }
1980     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1981   }
1982 } {
1983   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1984   \prop_put:Nnx \l_tmpa_prop { arity }
1985     { \str_count:N \l_stex_symdecl_args_str }
1986 }
1987 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1988
1989

```

```

1990 % semantic macro
1991
1992 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1993   \exp_args:Nx \stex_do_aftergroup:n {
1994     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1995       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1996     }}
1997   }
1998
1999   \bool_if:NF \l_stex_symdecl_local_bool {
2000     \exp_args:Nx \stex_add_to_current_module:n {
2001       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2002         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2003       } }
2004     }
2005   }
2006 }
2007
2008 % add to all symbols
2009
2010 \bool_if:NF \l_stex_symdecl_local_bool {
2011   \exp_args:Nx \stex_add_to_current_module:n {
2012     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2013       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2014     }
2015   }
2016 %   \exp_args:Nx \stex_add_field_to_current_module:n {
2017 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018 %   }
2019 }
2020
2021 \stex_debug:nn{symbols}{New~symbol:~
2022   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2023   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2024   Args:~\prop_item:Nn \l_tmpa_prop { args }
2025 }
2026
2027 % circular dependencies require this:
2028
2029 \prop_if_exist:cF {
2030   l_stex_symdecl_
2031   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2032   _prop
2033 } {
2034   \prop_set_eq:cN {
2035     l_stex_symdecl_
2036     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2037     _prop
2038   } \l_tmpa_prop
2039 }
2040
2041 \seq_clear:c {
2042   l_stex_symdecl_
2043   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2044   _notations
2045 }
2046
2047 \bool_if:NF \l_stex_symdecl_local_bool {
2048   \exp_args:Nx
2049   \stex_add_to_current_module:n {
2050     \seq_clear:c {
2051       l_stex_symdecl_
2052       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2053       _notations
2054     }
2055     \prop_set_from_keyval:cn {
2056       l_stex_symdecl_
2057       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2058       _prop
2059     } {
2060       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2061       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2062       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2063       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2064       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2065       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2066     }
2067   }
2068 }
2069
2070 \stex_if_smsmode:TF {
2071   \bool_if:NF \l_stex_symdecl_local_bool {
2072     % \exp_args:Nx \stex_add_to_sms:n {
2073     %   \prop_set_from_keyval:cn {
2074     %     l_stex_symdecl_
2075     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2076     %     _prop
2077     %   } {
2078     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2079     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2080     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2081     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2082     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2083     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2084     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2085     %   }
2086     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2087     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2088     %   }
2089     % }
2090   }
2091 }{
2092   \exp_args:Nx \stex_do_aftergroup:n {
2093     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2094       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2095     }
2096   }
2097   \stex_if_do_html:T {

```



```

2098 \stex_annotate_invisible:nnn {symdecl} {
2099   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2100 } {
2101   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2102   \stex_annotate_invisible:nnn{args}{}{
2103     \prop_item:Nn \l_tmpa_prop { args }
2104   }
2105   \stex_annotate_invisible:nnn{macroname}{#1}{}
2106   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2107     \stex_annotate_invisible:nnn{definiens}{}
2108     { $\l_stex_symdecl_definiens_tl$ }
2109   }
2110 }
2111 }
2112 }
2113 }

```

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

`\stex_get_symbol:n`

```

2114 \str_new:N \l_stex_get_symbol_uri_str
2115
2116 \cs_new_protected:Nn \stex_get_symbol:n {
2117   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2118     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2119   }{
2120     % argument is a string
2121     % is it a command name?
2122     \cs_if_exist:cTF { #1 }{
2123       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2124       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2125       \str_if_empty:NNTF \l_tmpa_str {
2126         \exp_args:Nx \cs_if_eq:NNTF {
2127           \tl_head:N \l_tmpa_tl
2128         } \stex_invoke_symbol:n {
2129           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2130         }{
2131           \__stex_symdecl_get_symbol_from_string:n { #1 }
2132         }
2133       } {
2134         \__stex_symdecl_get_symbol_from_string:n { #1 }
2135       }
2136     }{
2137       % argument is not a command name
2138       \__stex_symdecl_get_symbol_from_string:n { #1 }
2139       % \l_stex_all_symbols_seq
2140     }
2141   }
2142 }
2143
2144 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2145   \str_set:Nn \l_tmpa_str { #1 }
2146   \bool_set_false:N \l_tmpa_bool
2147   \stex_if_in_module:T {

```

```

2148 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2149 \bool_set_true:N \l_tmpa_bool
2150 \str_set:Nx \l_stex_get_symbol_uri_str {
2151 \l_stex_current_module_str ? #1
2152 }
2153 }
2154 }
2155 \bool_if:NF \l_tmpa_bool {
2156 \tl_set:Nn \l_tmpa_tl {
2157 \msg_set:nnn{stex}{error/unknownsymbol}{
2158 No~symbol~#1~found!
2159 }
2160 \msg_error:nn{stex}{error/unknownsymbol}
2161 }
2162 \str_set:Nn \l_tmpa_str { #1 }
2163 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2164 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2165 \str_set:Nn \l_tmpb_str { ##1 }
2166 \str_if_eq:eeT { \l_tmpa_str } {
2167 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2168 } {
2169 \seq_map_break:n {
2170 \tl_set:Nn \l_tmpa_tl {
2171 \str_set:Nn \l_stex_get_symbol_uri_str {
2172 ##1
2173 }
2174 }
2175 }
2176 }
2177 }
2178 \l_tmpa_tl
2179 }
2180 }
2181
2182 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2183 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2184 { \tl_tail:N \l_tmpa_tl }
2185 \tl_if_single:NTF \l_tmpa_tl {
2186 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2187 \exp_after:wN \str_set:Nn \exp_after:wN
2188 \l_stex_get_symbol_uri_str \l_tmpa_tl
2189 }{
2190 % TODO
2191 % tail is not a single group
2192 }
2193 }{
2194 % TODO
2195 % tail is not a single group
2196 }
2197 }

```

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

30.2 Notations

```

2198 <@@=stex_notation>

      notation arguments:
2199 \keys_define:nn { stex / notation } {
2200   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2201   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2202   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2203   op        .tl_set:N   = \l__stex_notation_op_tl ,
2204   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2205   primary   .default:n   = {true} ,
2206   unknown   .code:n      = \str_set:Nx
2207             \l__stex_notation_variant_str \l_keys_key_str
2208 }
2209
2210 \cs_new_protected:Nn \stex_notation_args:n {
2211   \str_clear:N \l__stex_notation_lang_str
2212   \str_clear:N \l__stex_notation_variant_str
2213   \str_clear:N \l__stex_notation_prec_str
2214   \tl_clear:N \l__stex_notation_op_tl
2215   \bool_set_false:N \l__stex_notation_primary_bool
2216
2217   \keys_set:nn { stex / notation } { #1 }
2218 }

```

\notation

```

2219 \NewDocumentCommand \notation { 0{ } m } {
2220   \stex_notation_args:n { #1 }
2221   \tl_clear:N \l_stex_symdecl_definiens_tl
2222   \stex_get_symbol:n { #2 }
2223   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2224 }
2225 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2226 \cs_new_protected:Nn \stex_notation_do:nn {
2227   \let\l_stex_current_symbol_str\relax
2228   \prop_set_eq:Nc \l_tmpa_prop {
2229     l_stex_symdecl_ #1 _prop
2230   }
2231
2232   \prop_clear:N \l_tmpb_prop
2233   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2234   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2235   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2236
2237   % precedences
2238   \seq_clear:N \l_tmpb_seq
2239   \exp_args:NNno
2240   \str_if_empty:NTF \l__stex_notation_prec_str {
2241     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2242     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2243     \exp_args:NNx
2244     \prop_put:Nno \l_tmpb_prop { opprec }
2245     { \neginfprec }
2246   }{
2247     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2248   }
2249 } {
2250   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2251     \exp_args:NNx
2252     \prop_put:Nno \l_tmpb_prop { opprec }
2253     { \neginfprec }
2254     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2255     \int_step_inline:nn { \l_tmpa_str } {
2256       \exp_args:NNx
2257       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2258     }
2259   }{
2260     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2261     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2262       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2263       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2264         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2265         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2266         \seq_map_inline:Nn \l_tmpa_seq {
2267           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2268         }
2269       }
2270       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2271     }{
2272       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2273       \int_compare:nNnTF \l_tmpa_str = 0 {
2274         \exp_args:NNx
2275         \prop_put:Nno \l_tmpb_prop { opprec }
2276         { \infprec }
2277       }{
2278         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2279       }
2280     }
2281   }
2282 }
2283
2284 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2285 \int_step_inline:nn { \l_tmpa_str } {
2286   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2287     \exp_args:NNx
2288     \seq_put_right:Nn \l_tmpb_seq {
2289       \prop_item:Nn \l_tmpb_prop { opprec }
2290     }
2291   }
2292 }
2293
2294 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2295 \tl_clear:N \l_tmpa_tl
2296

```

```

2297 \int_compare:nNnTF \l_tmpa_str = 0 {
2298   \exp_args:NNe
2299   \cs_set:Npn \l__stex_notation_macrocode_cs {
2300     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2301     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2302     { \prop_item:Nn \l_tmpb_prop { opprec } }
2303     { \exp_not:n { #2 } }
2304   }
2305   \__stex_notation_final:
2306 }{
2307   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2308   \str_if_in:NnTF \l_tmpb_str b {
2309     \exp_args:Nne \use:nn
2310     {
2311       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2312       \cs_set:Npn \l_tmpa_str } { {
2313         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2314         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2315         { \prop_item:Nn \l_tmpb_prop { opprec } }
2316         { \exp_not:n { #2 } }
2317       }}
2318     }{
2319       \str_if_in:NnTF \l_tmpb_str B {
2320         \exp_args:Nne \use:nn
2321         {
2322           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2323           \cs_set:Npn \l_tmpa_str } { {
2324             \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2325             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2326             { \prop_item:Nn \l_tmpb_prop { opprec } }
2327             { \exp_not:n { #2 } }
2328           } }
2329         }{
2330           \exp_args:Nne \use:nn
2331           {
2332             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2333             \cs_set:Npn \l_tmpa_str } { {
2334               \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2335               { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2336               { \prop_item:Nn \l_tmpb_prop { opprec } }
2337               { \exp_not:n { #2 } }
2338             } }
2339           }
2340         }
2341       \int_zero:N \l_tmpa_int
2342       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2343       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2344       \__stex_notation_arguments:
2345     }
2346   }
2347 }

```

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

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

```

2348 \cs_new_protected:Nn \_stex_notation_arguments: {
2349   \int_incr:N \l_tmpa_int
2350   \str_if_empty:NTF \l_tmpa_str {
2351     \_stex_notation_final:
2352   }{
2353     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2354     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2355     \str_if_eq:VnTF \l_tmpb_str a {
2356       \_stex_notation_argument_assoc:n
2357     }{
2358       \str_if_eq:VnTF \l_tmpb_str B {
2359         \_stex_notation_argument_assoc:n
2360       }{
2361         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2362         \tl_put_right:Nx \l_tmpa_tl {
2363           { \_stex_term_math_arg:nnn
2364             { \int_use:N \l_tmpa_int }
2365             { \l_tmpb_str }
2366             { ####\int_use:N \l_tmpa_int }
2367           }
2368         }
2369         \_stex_notation_arguments:
2370       }
2371     }
2372   }
2373 }
```

(End definition for `_stex_notation_arguments:.`)

`_stex_notation_argument_assoc:n`

```

2374 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2375   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2376   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2377   \tl_put_right:Nx \l_tmpa_tl {
2378     { \_stex_term_math_assoc_arg:nnnn
2379       { \int_use:N \l_tmpa_int }
2380       { \l_tmpb_str }
2381       \exp_args:No \exp_not:n
2382       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2383       { ####\int_use:N \l_tmpa_int }
2384     }
2385   }
2386   \_stex_notation_arguments:
2387 }
```

(End definition for `_stex_notation_argument_assoc:n.`)

`_stex_notation_final:` Called after processing all notation arguments

```

2388 \cs_new_protected:Nn \_stex_notation_final: {
2389   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2390   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2391   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2392   \exp_args:Nne \use:nn
```

```

2393 {
2394 \cs_generate_from_arg_count:cNnn {
2395   stex_notation_ \l_tmpa_str \c_hash_str
2396   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2397   _cs
2398 }
2399 \cs_set:Npn \l_tmpb_str } { {
2400   \exp_after:wN \exp_after:wN \exp_after:wN
2401   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2402   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2403 } }
2404
2405 \tl_if_empty:NF \l__stex_notation_op_tl {
2406   \cs_set:cpx {
2407     stex_op_notation_ \l_tmpa_str \c_hash_str
2408     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2409     _cs
2410   } {
2411     \_stex_term_oms:nnn {
2412       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2413       \l__stex_notation_lang_str
2414     }{
2415       \l_tmpa_str
2416     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2417   }
2418 }
2419
2420 \exp_args:Ne
2421 \stex_add_to_current_module:n {
2422   \cs_generate_from_arg_count:cNnn {
2423     stex_notation_ \l_tmpa_str \c_hash_str
2424     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2425     _cs
2426   } \cs_set:Npn {\l_tmpb_str} {
2427     \exp_after:wN \exp_after:wN \exp_after:wN
2428     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2429     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2430   }
2431   \tl_if_empty:NF \l__stex_notation_op_tl {
2432     \cs_set:cpn {
2433       stex_op_notation_ \l_tmpa_str \c_hash_str
2434       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2435       _cs
2436     } {
2437       \_stex_term_oms:nnn {
2438         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2439         \l__stex_notation_lang_str
2440       }{
2441         \l_tmpa_str
2442       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2443     }
2444   }
2445 }
2446

```

```

2447 \seq_put_right:cx {
2448   l_stex_symdecl_
2449   \prop_item:Nn \l_tmpb_prop { symbol }
2450   _notations
2451 } {
2452   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2453 }
2454
2455 \stex_debug:nn{symbols}{
2456   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2457   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2458   Operator~precedence:~
2459   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2460   Argument~precedences:~
2461   \seq_use:Nn \l_tmpa_seq {,~}^^J
2462   Notation: \cs_meaning:c {
2463     stex_notation_ \l_tmpa_str \c_hash_str
2464     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2465     _cs
2466   }
2467 }
2468
2469 \prop_set_eq:cN {
2470   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2471   \c_hash_str \l__stex_notation_lang_str _prop
2472 } \l_tmpb_prop
2473
2474 \exp_args:Ne
2475 \stex_add_to_current_module:n {
2476   \seq_put_right:cn {
2477     l_stex_symdecl_
2478     \prop_item:Nn \l_tmpb_prop { symbol }
2479     _notations
2480   } {
2481     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2482   }
2483   \prop_set_from_keyval:cn {
2484     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2485     \c_hash_str \l__stex_notation_lang_str _prop
2486   } {
2487     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2488     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2489     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2490     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2491     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2492   }
2493 }
2494
2495 \stex_if_smsmode:TF {
2496   \stex_smsmode_set_codes:
2497   % \exp_args:Nx \stex_add_to_sms:n {
2498   %   \prop_set_from_keyval:cn {
2499   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2500   %     \c_hash_str \l__stex_notation_lang_str _prop

```



```

2501 %      } {
2502 %          symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2503 %          language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2504 %          variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2505 %          opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2506 %          argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2507 %      }
2508 %  }
2509 }{
2510
2511 % HTML annotations
2512 \stex_if_do_html:T {
2513     \stex_annotate_invisible:nnn { notation }
2514     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2515         \stex_annotate_invisible:nnn { notationfragment }
2516         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2517         \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2518         \stex_annotate_invisible:nnn { precedence }
2519         { \prop_item:Nn \l_tmpb_prop { opprec };
2520           \seq_use:Nn \l_tmpa_seq { x }
2521         }{}
2522
2523         \int_zero:N \l_tmpa_int
2524         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2525         \tl_clear:N \l_tmpa_tl
2526         \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2527             \int_incr:N \l_tmpa_int
2528             \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2529             \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2530             \str_if_eq:VnTF \l_tmpb_str a {
2531                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2532                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2533                     \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2534                 } }
2535             }{
2536                 \str_if_eq:VnTF \l_tmpb_str B {
2537                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2538                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2539                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2540                     } }
2541                 }{
2542                     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2543                         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2544                     } }
2545                 }
2546             }
2547         }
2548         \stex_annotate_invisible:nnn { notationcomp }{}{
2549             \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2550             $ \exp_args:Nno \use:nn { \use:c {
2551                 stex_notation_ \l_stex_current_symbol_str
2552                 \c_hash_str \l__stex_notation_variant_str
2553                 \c_hash_str \l__stex_notation_lang_str _cs
2554             } } { \l_tmpa_tl } $

```

```

2555     }
2556   }
2557 }
2558 }
2559 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2560 \keys_define:nn { stex / setnotation } {
2561   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2562   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2563   unknown .code:n = \str_set:Nx
2564     \l__stex_notation_variant_str \l_keys_key_str
2565 }
2566
2567 \cs_new_protected:Nn \_stex_setnotation_args:n {
2568   \str_clear:N \l__stex_notation_lang_str
2569   \str_clear:N \l__stex_notation_variant_str
2570   \keys_set:nn { stex / setnotation } { #1 }
2571 }
2572
2573 \NewDocumentCommand \setnotation {m m} {
2574   \stex_get_symbol:n { #1 }
2575   \_stex_setnotation_args:n { #2 }
2576   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2577     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2578     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2579       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2580     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2581       { \c_hash_str }
2582     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2583       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2584     \exp_args:Nx \stex_add_to_current_module:n {
2585       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2586         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2587     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2588       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2589     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2590       { \c_hash_str }
2591   }
2592   \stex_debug:nn {notations}{
2593     Setting~default~notation~
2594     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2595     \l_stex_get_symbol_uri_str \
2596     \expandafter\meaning\csname
2597     l_stex_symdecl \l_stex_get_symbol_uri_str _notations\endcsname
2598   }
2599   }{
2600     % todo throw error
2601   }
2602 }
2603

```

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

\symdef

```
2604 \keys_define:nn { stex / symdef } {
2605   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2606   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2607   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2608   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2609   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2610   op        .tl_set:N    = \l__stex_notation_op_tl ,
2611   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2612   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2613   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2614   unknown   .code:n      = \str_set:Nx
2615             \l__stex_notation_variant_str \l_keys_key_str
2616 }
2617
2618 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2619   \str_clear:N \l_stex_symdecl_name_str
2620   \str_clear:N \l_stex_symdecl_args_str
2621   \bool_set_false:N \l_stex_symdecl_local_bool
2622   \tl_clear:N \l_stex_symdecl_type_tl
2623   \tl_clear:N \l_stex_symdecl_definiens_tl
2624   \str_clear:N \l__stex_notation_lang_str
2625   \str_clear:N \l__stex_notation_variant_str
2626   \str_clear:N \l__stex_notation_prec_str
2627   \tl_clear:N \l__stex_notation_op_tl
2628
2629   \keys_set:nn { stex / symdef } { #1 }
2630 }
2631
2632 \NewDocumentCommand \symdef { 0{} m } {
2633   \__stex_notation_symdef_args:n { #1 }
2634   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2635   \stex_symdecl_do:n { #2 }
2636   \exp_args:Nx \stex_notation_do:nn {
2637     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2638   }
2639 }
2640 \stex_deactivate_macro:Nn \symdef {module~environments}
2641
2642 (End definition for \symdef. This function is documented on page 37.)
2643 \endpackage
```

Chapter 31

STEX -Terms Implementation

```
2642 <*package>
2643
2644 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2645
2646 <@@=stex_terms>
2647
2648 Warnings and error messages
2649 \msg_new:nnn{stex}{error/nonotation}{
2650   Symbol~#1~invoked,~but~has~no~notation~#2!
2651 }
2652 \msg_new:nnn{stex}{error/notationarg}{
2653   Error~in~parsing~notation~#1
2654 }
2655 \msg_new:nnn{stex}{error/noop}{
2656   Symbol~#1~has~no~operator~notation~for~notation~#2
2657 }
```

31.1 Symbol Invocations

Arguments:

```
2657 \keys_define:nn { stex / terms } {
2658   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2659   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2660   unknown .code:n = \str_set:Nx
2661     \l__stex_terms_variant_str \l_keys_key_str
2662 }
2663
2664 \cs_new_protected:Nn \__stex_terms_args:n {
2665   \str_clear:N \l__stex_terms_lang_str
2666   \str_clear:N \l__stex_terms_variant_str
2667   \str_clear:N \l__stex_terms_prec_str
2668   \tl_clear:N \l__stex_terms_op_tl
2669
2670   \keys_set:nn { stex / terms } { #1 }
```

2671 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2672 \cs_new_protected:Nn \stex_invoke_symbol:n {
2673   \if_mode_math:
2674     \exp_after:wN \__stex_terms_invoke_math:n
2675   \else:
2676     \exp_after:wN \__stex_terms_invoke_text:n
2677   \fi: { #1 }
2678 }
```

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

__stex_terms_invoke_math:n

```
2679 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2680   \peek_charcode_remove:NTF ! {
2681     \peek_charcode:NTF [ {
2682       \__stex_terms_invoke_op:nw { #1 }
2683     }{
2684       \peek_charcode_remove:NTF ! {
2685         \peek_charcode:NTF [ {
2686           \__stex_terms_invoke_op_custom:nw
2687         }{
2688           % TODO throw error
2689         }
2690       }{
2691         \__stex_terms_invoke_op:nw { #1 } []
2692       }
2693     }
2694   }{
2695     \peek_charcode_remove:NTF * {
2696       \__stex_terms_invoke_text:n { #1 }
2697     }{
2698       \peek_charcode:NTF [ {
2699         \__stex_terms_invoke_math:nw { #1 }
2700       }{
2701         \__stex_terms_invoke_math:nw { #1 } []
2702       }
2703     }
2704   }
2705 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2706 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2707   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2708     \stex_highlight_term:nn{#1}{#2}
2709   }
2710 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

__stex_terms_invoke_op:nw

```

2711 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2712   \__stex_terms_args:n { #2 }
2713   \cs_if_exist:cTF {
2714     stex_op_notation_ #1 \c_hash_str
2715     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2716   }{
2717     \csname stex_op_notation_ #1 \c_hash_str
2718       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2719     \endcsname
2720   }{
2721     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2722   }
2723 }

```

(End definition for __stex_terms_invoke_op:nw.)

__stex_terms_invoke_math:nw

```

2724 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2725   \__stex_terms_args:n { #2 }
2726   \seq_if_empty:cTF {
2727     l_stex_symdecl_ #1 _notations
2728   } {
2729     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2730   } {
2731     \seq_if_in:cxTF {
2732       l_stex_symdecl_ #1 _notations
2733     }
2734     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2735       \str_set:Nn \l_stex_current_symbol_str { #1 }
2736       \use:c{
2737         stex_notation_ #1 \c_hash_str
2738         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2739         _cs
2740       }
2741     }{
2742       \str_if_empty:NTF \l__stex_terms_variant_str {
2743         \str_if_empty:NTF \l__stex_terms_lang_str {
2744           \seq_get_left:cN {
2745             l_stex_symdecl_ #1 _notations
2746           } \l_tmpa_str
2747           \str_set:Nn \l_stex_current_symbol_str { #1 }
2748           \use:c{
2749             stex_notation_ #1 \c_hash_str \l_tmpa_str
2750             _cs
2751           }
2752         }{
2753           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2754             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2755           }
2756         }
2757       }{
2758         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2759           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2760     }
2761   }
2762 }
2763 }
2764 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2765 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2766   \peek_charcode_remove:NTF ! {
2767     \stex_term_custom:nn { #1 } { }
2768   }{
2769     \prop_set_eq:Nc \l_tmpa_prop {
2770       l_stex_symdecl_ #1 _prop
2771     }
2772     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2773     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2774   }
2775 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2776 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2777 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2778 \int_new:N \l__stex_terms_downprec
2779 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 39.)

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2780 \tl_set:Nn \l__stex_terms_left_bracket_str (
2781 \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

```

2782 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2783   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2784     \bool_set_false:N \l__stex_terms_brackets_done_bool
2785     #2
2786   } {
2787     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2788       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2789         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2790         \dobrackets { #2 }
2791       }

```

```

2792     }{ #2 }
2793   }
2794 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2795 \bool_new:N \l__stex_terms_brackets_done_bool
2796 %\RequirePackage{scalerel}
2797 \cs_new_protected:Npn \dobrackets #1 {
2798   %\ThisStyle{\if D\m@switch
2799   %   \exp_args:Nnx \use:nn
2800   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2801   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2802   % \else
2803   \exp_args:Nnx \use:nn
2804   {
2805     \bool_set_true:N \l__stex_terms_brackets_done_bool
2806     \int_set:Nn \l__stex_terms_downprec \infprec
2807     \l__stex_terms_left_bracket_str
2808     #1
2809   }
2810   {
2811     \bool_set_false:N \l__stex_terms_brackets_done_bool
2812     \l__stex_terms_right_bracket_str
2813     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2814   }
2815   %\fi}
2816 }

```

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

`\withbrackets`

```

2817 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2818   \exp_args:Nnx \use:nn
2819   {
2820     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2821     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2822     #3
2823   }
2824   {
2825     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2826     {\l__stex_terms_left_bracket_str}
2827     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2828     {\l__stex_terms_right_bracket_str}
2829   }
2830 }

```

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

`\STEXinvisible`

```

2831 \cs_new_protected:Npn \STEXinvisible #1 {
2832   \stex_annotate_invisible:n { #1 }
2833 }

```


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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2834 \cs_new_protected:Nn \_stex_term_oms:nnn {
2835   \stex_annotate:nnn{ OMID }{ #2 }{
2836     \stex_highlight_term:nn { #1 } { #3 }
2837   }
2838 }
2839
2840 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2841   \_stex_terms_maybe_brackets:nn { #3 }{
2842     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2843   }
2844 }

```

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

`_stex_term_math_oma:nnnn`

```

2845 \cs_new_protected:Nn \_stex_term_oma:nnn {
2846   \stex_annotate:nnn{ OMA }{ #2 }{
2847     \stex_highlight_term:nn { #1 } { #3 }
2848   }
2849 }
2850
2851 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2852   \_stex_terms_maybe_brackets:nn { #3 }{
2853     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2854   }
2855 }

```

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

`_stex_term_math_omb:nnnn`

```

2856 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2857   \stex_annotate:nnn{ OMBIND }{ #2 }{
2858     \stex_highlight_term:nn { #1 } { #3 }
2859   }
2860 }
2861
2862 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2863   \_stex_terms_maybe_brackets:nn { #3 }{
2864     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2865   }
2866 }

```

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

`_stex_term_math_arg:nnn`

```

2867 \cs_new_protected:Nn \_stex_term_arg:nn {
2868   \stex_unhighlight_term:n {
2869     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2870   }
2871 }

```

```

2872 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2873   \exp_args:Nnx \use:nn
2874     { \int_set:Nn \l__stex_terms_downprec { #2 }
2875       \stex_term_arg:nn { #1 }{ #3 }
2876     }
2877     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2878   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2879 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2880   \clist_set:Nn \l_tmpa_clist{ #4 }
2881   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2882     \tl_set:Nn \l_tmpa_tl { #4 }
2883   }{
2884     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2885     \clist_reverse:N \l_tmpa_clist
2886     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2887
2888     \clist_map_inline:Nn \l_tmpa_clist {
2889       \exp_args:NNNo \exp_args:Nno \tl_set:Nn \l_tmpa_tl {
2890         \exp_args:Nno
2891           \l_tmpa_cs { ##1 } \l_tmpa_tl
2892       }
2893     }
2894
2895   }
2896   \exp_args:Nnno
2897   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2898 }

```

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

`\stex_term_custom:nn`

```

2899 \cs_new_protected:Nn \stex_term_custom:nn {
2900   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2901   \str_set:Nn \l_tmpa_str { #2 }
2902   \tl_clear:N \l_tmpa_tl
2903   \int_zero:N \l_tmpa_int
2904   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2905   \__stex_terms_custom_loop:
2906 }

```

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

`__stex_terms_custom_loop:`

```

2907 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2908   \bool_set_false:N \l_tmpa_bool
2909   \bool_while_do:nn {
2910     \str_if_eq_p:ee X {
2911       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2912     }
2913   }{
2914     \int_incr:N \l_tmpa_int

```

```

2915 }
2916
2917 \peek_charcode:NTF [ {
2918   % notation/text component
2919   \__stex_terms_custom_component:w
2920 } {
2921   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2922     % all arguments read => finish
2923     \__stex_terms_custom_final:
2924   } {
2925     % arguments missing
2926     \peek_charcode_remove:NTF * {
2927       % invisible, specific argument position or both
2928       \peek_charcode:NTF [ {
2929         % visible specific argument position
2930         \__stex_terms_custom_arg:wn
2931       } {
2932         % invisible
2933         \peek_charcode_remove:NTF * {
2934           % invisible specific argument position
2935           \__stex_terms_custom_arg_inv:wn
2936         } {
2937           % invisible next argument
2938           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2939         }
2940       }
2941     } {
2942       % next normal argument
2943       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2944     }
2945   }
2946 }
2947 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2948 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2949   \bool_set_true:N \l_tmpa_bool
2950   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2951 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2952 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2953   \str_set:Nx \l_tmpb_str {
2954     \str_item:Nn \l_tmpa_str { #1 }
2955   }
2956   \str_case:VnTF \l_tmpb_str {
2957     { X } {
2958       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2959     }
2960     { i } { \__stex_terms_custom_set_X:n { #1 } }
2961     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2962     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2963     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2964   }{}{
2965     \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2966   }
2967
2968   \bool_if:nTF \l_tmpa_bool {
2969     \tl_put_right:Nx \l_tmpa_tl {
2970       \stex_annotate_invisible:n {
2971         \stex_term_arg:nn { \int_eval:n { #1 } }
2972         \exp_not:n { { #2 } }
2973       }
2974     }
2975   } {
2976     \tl_put_right:Nx \l_tmpa_tl {
2977       \stex_term_arg:nn { \int_eval:n { #1 } }
2978       \exp_not:n { { #2 } }
2979     }
2980   }
2981
2982   \_stex_terms_custom_loop:
2983 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2984 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2985   \str_set:Nx \l_tmpa_str {
2986     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2987     X
2988     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2989   }
2990 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2991 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2992   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2993   \_stex_terms_custom_loop:
2994 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2995 \cs_new_protected:Nn \_stex_terms_custom_final: {
2996   \int_compare:nNnTF \l_tmpb_int = 0 {
2997     \exp_args:Nnno \stex_term_oms:nnn
2998   }{
2999     \str_if_in:NnTF \l_tmpa_str {b} {
3000       \exp_args:Nnno \stex_term_ombind:nnn
3001     } {
3002       \exp_args:Nnno \stex_term_oma:nnn
3003     }
3004   }

```

```

3005 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3006 }

```

(End definition for `_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

3007 \NewDocumentCommand \symref { m m }{
3008   \let\compemph_uri_prev:\compemph@uri
3009   \let\compemph@uri\symrefemph@uri
3010   \STEXsymbol{#1}! [#2]
3011   \let\compemph@uri\compemph_uri_prev:
3012 }
3013
3014 \keys_define:nn { stex / symname } {
3015   post      .str_set_x:N    = \l_stex_symname_post_str
3016 }
3017
3018 \cs_new_protected:Nn \stex_symname_args:n {
3019   \str_clear:N \l_stex_symname_post_str
3020   \keys_set:nn { stex / symname } { #1 }
3021 }
3022
3023 \NewDocumentCommand \symname { O{} m }{
3024   \stex_symname_args:n { #1 }
3025   \stex_get_symbol:n { #2 }
3026   \str_set:Nx \l_tmpa_str {
3027     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3028   }
3029   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3030
3031   \let\compemph_uri_prev:\compemph@uri
3032   \let\compemph@uri\symrefemph@uri
3033   \exp_args:NNx \use:nn
3034   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3035     \l_tmpa_str \l_stex_symname_post_str
3036   ] }
3037   \let\compemph@uri\compemph_uri_prev:
3038 }

```

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

31.3 Notation Components

```

3039 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3040
3041 \str_new:N \l_stex_current_symbol_str
3042 \cs_new_protected:Nn \stex_highlight_term:nn {
3043   \exp_args:Nnx
3044   \use:nn {
3045     \str_set:Nx \l_stex_current_symbol_str { #1 }
3046     #2
3047   } {

```

```

3048 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3049 { \l_stex_current_symbol_str }
3050 }
3051 }
3052
3053 \cs_new_protected:Nn \stex_unhighlight_term:n {
3054 % \latexml_if:TF {
3055 % #1
3056 % } {
3057 % \rustex_if:TF {
3058 % #1
3059 % } {
3060 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3061 % }
3062 % }
3063 }

```

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

```

\comp
\compemph@uri 3064 \cs_new_protected:Npn \comp #1 {
\compemph 3065 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3066 \rustex_if:TF {
\defemph@uri 3067 \stex_annotate:nnn { comp }{\l_stex_current_symbol_str}{ #1 }
\symrefemph 3068 }{
\symrefemph@uri 3069 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3070 }
3071 }
3072 }
3073
3074 \cs_new_protected:Npn \compemph@uri #1 #2 {
3075 \compemph{ #1 }
3076 }
3077
3078
3079 \cs_new_protected:Npn \compemph #1 {
3080 #1
3081 }
3082
3083 \cs_new_protected:Npn \defemph@uri #1 #2 {
3084 \defemph{#1}
3085 }
3086
3087 \cs_new_protected:Npn \defemph #1 {
3088 \textbf{#1}
3089 }
3090
3091 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3092 \symrefemph{#1}
3093 }
3094
3095 \cs_new_protected:Npn \symrefemph #1 {
3096 \textbf{#1}
3097 }

```

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

`\ellipses`

```
3098 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```
\parray
\prmatrix 3099 \bool_new:N \l_stex_inarray_bool
\parrayline 3100 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3101 \NewDocumentCommand \parray { m m } {
\parraycell 3102 \begingroup
3103 \bool_set_true:N \l_stex_inarray_bool
3104 \begin{array}{#1}
3105 #2
3106 \end{array}
3107 \endgroup
3108 }
3109
3110 \NewDocumentCommand \prmatrix { m } {
3111 \begingroup
3112 \bool_set_true:N \l_stex_inarray_bool
3113 \begin{matrix}
3114 #1
3115 \end{matrix}
3116 \endgroup
3117 }
3118
3119 \def \maybepline {
3120 \bool_if:NT \l_stex_inarray_bool {\hline}
3121 }
3122
3123 \def \parrayline #1 #2 {
3124 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3125 }
3126
3127 \def \pmrow #1 { \parrayline{}{ #1 } }
3128
3129 \def \parraylineh #1 #2 {
3130 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3131 }
3132
3133 \def \parraycell #1 {
3134 #1 \bool_if:NT \l_stex_inarray_bool {&}
3135 }
```

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

```
3136 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
3137 <*package>
3138
3139 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3140
3141 <@@=stex_features>
3142
3143   Warnings and error messages
3144   \msg_new:nnn{stex}{error/copymodule/notallowed}{
3145     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3146   }
3147   \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3148     Symbol~#1~not~assigned~in~interpretmodule~#2
3149   }
```

32.1 Imports with modification

```
3149 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3150   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3151     \__stex_features_get_symbol_from_cs:n { #1 }
3152   }{
3153     % argument is a string
3154     % is it a command name?
3155     \cs_if_exist:cTF { #1 }{
3156       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3157       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3158       \str_if_empty:NNTF \l_tmpa_str {
3159         \exp_args:Nx \cs_if_eq:NNTF {
3160           \tl_head:N \l_tmpa_tl
3161         } \stex_invoke_symbol:n {
3162           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3163         }{
3164           \__stex_features_get_symbol_from_string:n { #1 }
```



```

3165     }
3166   } {
3167     \__stex_features_get_symbol_from_string:n { #1 }
3168   }
3169   ){
3170     % argument is not a command name
3171     \__stex_features_get_symbol_from_string:n { #1 }
3172     % \l_stex_all_symbols_seq
3173   }
3174 }
3175 }
3176
3177 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3178   \str_set:Nn \l_tmpa_str { #1 }
3179   \bool_set_false:N \l_tmpa_bool
3180   \bool_if:NF \l_tmpa_bool {
3181     \tl_set:Nn \l_tmpa_tl {
3182       \msg_set:nnn{stex}{error/unknownsymbol}{
3183         No~symbol~#1~found!
3184       }
3185       \msg_error:nn{stex}{error/unknownsymbol}
3186     }
3187     \str_set:Nn \l_tmpa_str { #1 }
3188     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3189     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3190       \str_set:Nn \l_tmpb_str { ##1 }
3191       \str_if_eq:eeT { \l_tmpa_str } {
3192         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3193       } {
3194         \seq_map_break:n {
3195           \tl_set:Nn \l_tmpa_tl {
3196             \str_set:Nn \l_stex_get_symbol_uri_str {
3197               ##1
3198             }
3199             \__stex_features_get_symbol_check:
3200           }
3201         }
3202       }
3203     }
3204     \l_tmpa_tl
3205   }
3206 }
3207
3208 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3209   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3210   { \tl_tail:N \l_tmpa_tl }
3211   \tl_if_single:NTF \l_tmpa_tl {
3212     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3213       \exp_after:wN \str_set:Nn \exp_after:wN
3214       \l_stex_get_symbol_uri_str \l_tmpa_tl
3215       \__stex_features_get_symbol_check:
3216     }{
3217       % TODO
3218       % tail is not a single group

```

```

3219     }
3220 }{
3221     % TODO
3222     % tail is not a single group
3223 }
3224 }
3225
3226 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3227     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3228     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3229         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3230         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3231         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3232             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3233                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3234             }
3235         }
3236     }{
3237         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3238             \l_stex_current_copymodule_name_str~(inexplicably)
3239         }
3240     }
3241 }
3242
3243 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3244     \stex_import_module_uri:nn { #1 } { #2 }
3245     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3246     \stex_import_require_module:nnnn
3247     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3248     { \l_stex_import_path_str } { \l_stex_import_name_str }
3249     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3250     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3251     \seq_clear:N \l__stex_features_copymodule_fields_seq
3252     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3253         \seq_map_inline:cn {c_stex_module_###1_constants}{
3254             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3255                 ###1 ? #####1
3256             }
3257         }
3258     }
3259     \seq_clear:N \l_tmpa_seq
3260     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3261         name      = \l_stex_current_copymodule_name_str ,
3262         module     = \l_stex_current_module_str ,
3263         from       = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3264         includes   = \l_tmpa_seq ,
3265         fields     = \l_tmpa_seq
3266     }
3267     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3268         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3269     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3270     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3271     \stex_if_smsmode:TF {
3272         \stex_smsmode_set_codes:

```

```

3273 } {
3274   \begin{stex_annotate_env} {#4} {
3275     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3276   }
3277   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3278 }
3279 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3280 \bool_set_false:N \l_stex_html_do_output_bool
3281 }
3282 \cs_new_protected:Nn \stex_copymodule_end:n {
3283   \def \l_tmpa_cs ##1 ##2 {#1}
3284   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3285   \tl_clear:N \l_tmpa_tl
3286   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3287   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3288     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3289       \l_tmpa_cs{##1}{####1}
3290       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3291         \tl_put_right:Nx \l_tmpa_tl {
3292           \prop_set_from_keyval:cn {
3293             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3294           }{
3295             \exp_after:wN \prop_to_keyval:N \csname
3296               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3297             \endcsname
3298           }
3299           \seq_clear:c {
3300             l_stex_symdecl_
3301             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3302             _notations
3303           }
3304         }
3305         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_name
3306         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3307         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3308           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3309           \tl_put_right:Nx \l_tmpa_tl {
3310             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3311               \stex_invoke_symbol:n {
3312                 \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3313               }
3314             }
3315           }
3316         }
3317       }{
3318         \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
3319         \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3320         \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3321         \tl_put_right:Nx \l_tmpa_tl {
3322           \prop_set_from_keyval:cn {
3323             l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3324           }{
3325             \prop_to_keyval:N \l_tmpa_prop
3326           }

```

```

3327         \seq_clear:c {
3328             l_stex_symdecl_
3329             \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3330             _notations
3331         }
3332     }
3333     \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1}
3334     \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3335         \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3336         \tl_put_right:Nx \l_tmpa_tl {
3337             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3338                 \stex_invoke_symbol:n {
3339                     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3340                 }
3341             }
3342         }
3343     }
3344 }
3345 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3346     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3347 }
3348 % todo notations
3349 }}
3350 }
3351 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3352 \tl_put_left:Nx \l_tmpa_tl {
3353     \prop_set_from_keyval:cn {
3354         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3355     }{
3356         \prop_to_keyval:N \l_stex_current_copymodule_prop
3357     }
3358 }
3359 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3360 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3361 \exp_args:Nx \stex_do_aftergroup:n {
3362     \exp_args:No \exp_not:n \l_tmpa_tl
3363 }
3364 \stex_if_smsmode:F {
3365     \end{stex_annotate_env}
3366 }
3367 }
3368
3369 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3370     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3371     \stex_deactivate_macro:Nn \symdecl {module~environments}
3372     \stex_deactivate_macro:Nn \symdef {module~environments}
3373     \stex_deactivate_macro:Nn \notation {module~environments}
3374     \stex_reactivate_macro:N \assign
3375     \stex_reactivate_macro:N \renamdecl
3376     \stex_reactivate_macro:N \donotcopy
3377 }{
3378     \stex_copymodule_end:n {}
3379 }
3380

```

```

3381 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3382   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3383   \stex_deactivate_macro:Nn \symdecl {module~environments}
3384   \stex_deactivate_macro:Nn \symdef {module~environments}
3385   \stex_deactivate_macro:Nn \notation {module~environments}
3386   \stex_reactivate_macro:N \assign
3387   \stex_reactivate_macro:N \renamedekl
3388   \stex_reactivate_macro:N \donotcopy
3389 }{
3390   \stex_copymodule_end:n {
3391     \tl_if_exist:cF {
3392       l__stex_features_copymodule_##1?##2_def_tl
3393     }{
3394       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3395         ##1?##2
3396       }{\l_stex_current_copymodule_name_str}
3397     }
3398   }
3399 }
3400
3401 \NewDocumentCommand \donotcopy { 0{} m}{
3402   \stex_import_module_uri:nn { #1 } { #2 }
3403   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3404   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3405     \seq_remove_all:Nn \l__stex_features_copymodule_modules_seq { ##1 }
3406     \seq_map_inline:cn {c_stex_module_##1_constants}{
3407       \seq_remove_all:Nn \l__stex_features_copymodule_fields_seq { ##1 ? #####1 }
3408       \bool_lazy_any_p:nT {
3409         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3410         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3411         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3412       }{
3413         % TODO throw error
3414       }
3415     }
3416   }
3417
3418   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3419   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3420   \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3421 }
3422
3423 \NewDocumentCommand \assign { m m }{
3424   \stex_get_symbol_in_copymodule:n {#1}
3425   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3426   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3427 }
3428
3429 \keys_define:nn { stex / renamedekl } {
3430   name .str_set_x:N = \l_stex_renamedekl_name_str
3431 }
3432 \cs_new_protected:Nn \__stex_features_renamedekl_args:n {
3433   \str_clear:N \l_stex_renamedekl_name_str
3434 }

```

```

3435 \keys_set:nn { stex / renamedecl } { #1 }
3436 }
3437
3438 \NewDocumentCommand \renamedecl { 0{} m m}{
3439 \__stex_features_renamedecl_args:n { #1 }
3440 \stex_get_symbol_in_copymodule:n {#2}
3441 \stex_debug:nn{renamedecl}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3442 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3443 \str_if_empty:NTF \l_stex_renamedecl_name_str {
3444 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3445 \l_stex_get_symbol_uri_str
3446 } }
3447 } {
3448 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3449 \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3450 \prop_set_eq:cc {l_stex_symdecl_
3451 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3452 _prop
3453 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3454 \seq_set_eq:cc {l_stex_symdecl_
3455 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3456 _notations
3457 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3458 \prop_put:cnx {l_stex_symdecl_
3459 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3460 _prop
3461 }{ name }{ \l_stex_renamedecl_name_str }
3462 \prop_put:cnx {l_stex_symdecl_
3463 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3464 _prop
3465 }{ module }{ \l_stex_current_module_str }
3466 \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3467 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3468 }
3469 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3470 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3471 } }
3472 }
3473 }
3474 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3475 % \_stex_notation_args:n { #1 }
3476 % \tl_clear:N \l_stex_symdecl_definiens_tl
3477 % \stex_get_symbol_in_copymodule:n { #2 }
3478 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3479 % % todo
3480 %}
3481 \stex_deactivate_macro:Nn \assign {copymodules}
3482 \stex_deactivate_macro:Nn \renamedecl {copymodules}
3483 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3484
3485
3486 \seq_new:N \l_stex_implicit_morphisms_seq
3487 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3488 \stex_import_module_uri:nn { #1 } { #2 }

```

```

3489 \stex_debug:nn{implicits}{
3490   Implicit~morphism:~
3491   \l_stex_module_ns_str ? \l__stex_features_name_str
3492 }
3493 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3494   \l_stex_module_ns_str ? \l__stex_features_name_str
3495 }{
3496   \msg_error:nnn{stex}{error/conflictingmodules}{
3497     \l_stex_module_ns_str ? \l__stex_features_name_str
3498   }
3499 }
3500
3501 % TODO
3502
3503
3504
3505 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3506   \l_stex_module_ns_str ? \l__stex_features_name_str
3507 }
3508 }
3509

```

32.2 The feature environment

structural@feature

```

3510
3511 \NewDocumentEnvironment{structural@feature}{ m m m }{
3512   \stex_if_in_module:F {
3513     \msg_set:nnn{stex}{error/nomodule}{
3514       Structural~Feature~has~to~occur~in~a~module:\\
3515       Feature~#2~of~type~#1\\
3516       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3517     }
3518     \msg_error:nn{stex}{error/nomodule}
3519   }
3520
3521   \str_set:Nx \l_stex_module_name_str {
3522     \prop_item:Nn \l_stex_current_module_prop
3523     { name } / #2 - feature
3524   }
3525
3526   \str_set:Nx \l_stex_module_ns_str {
3527     \prop_item:Nn \l_stex_current_module_prop
3528     { ns }
3529   }
3530
3531
3532   \str_clear:N \l_tmpa_str
3533   \seq_clear:N \l_tmpa_seq
3534   \tl_clear:N \l_tmpa_tl
3535   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3536     origname = #2,
3537     name      = \l_stex_module_name_str ,
3538     ns        = \l_stex_module_ns_str ,

```

```

3539 imports = \exp_not:o { \l_tmpa_seq } ,
3540 constants = \exp_not:o { \l_tmpa_seq } ,
3541 content = \exp_not:o { \l_tmpa_tl } ,
3542 file = \exp_not:o { \g_stex_currentfile_seq } ,
3543 lang = \l_stex_module_lang_str ,
3544 sig = \l_tmpa_str ,
3545 meta = \l_tmpa_str ,
3546 feature = #1 ,
3547 }
3548
3549 \stex_if_smsmode:TF {
3550   \stex_smsmode_set_codes:
3551 } {
3552   \begin{stex_annotate_env}{ feature:#1 }{}
3553   \stex_annotate_invisible:nnn{header}{}{ #3 }
3554 }
3555 }{
3556   \str_set:Nx \l_tmpa_str {
3557     c_stex_feature_
3558     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3559     \prop_item:Nn \l_stex_current_module_prop { name }
3560     _prop
3561   }
3562   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3563   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3564   \stex_if_smsmode:TF {
3565     \exp_args:Nx \stex_add_to_sms:n {
3566       \prop_gset_from_keyval:cn {
3567         c_stex_feature_
3568         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3569         \prop_item:Nn \l_stex_current_module_prop { name }
3570         _prop
3571       } {
3572         origname = #2,
3573         name = \prop_item:cn { \l_tmpa_str } { name } ,
3574         ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3575         imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3576         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3577         content = \prop_item:cn { \l_tmpa_str } { content } ,
3578         file = \prop_item:cn { \l_tmpa_str } { file } ,
3579         lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3580         sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3581         meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3582         feature = \prop_item:cn { \l_tmpa_str } { feature }
3583       }
3584     }
3585   } {
3586     \end{stex_annotate_env}
3587   }
3588 }
3589

```


32.3 Features

structure

```

3590
3591 \prop_new:N \l_stex_all_structures_prop
3592
3593 \keys_define:nn { stex / features / structure } {
3594   name .str_set_x:N = \l__stex_features_structure_name_str ,
3595 }
3596
3597 \cs_new_protected:Nn \__stex_features_structure_args:n {
3598   \str_clear:N \l__stex_features_structure_name_str
3599   \keys_set:nn { stex / features / structure } { #1 }
3600 }
3601
3602 %\stex_new_feature:nnnn { structure } { 0{} m } {
3603   % \__stex_features_structure_args:n { ##1 }
3604   % \str_if_empty:NT \l__stex_features_structure_name_str {
3605     % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3606   % }
3607   %} {
3608   %
3609   %}
3610
3611 \NewDocumentEnvironment{mathstructure}{ 0{} m }{
3612   \__stex_features_structure_args:n { #1 }
3613   \str_if_empty:NT \l__stex_features_structure_name_str {
3614     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3615   }
3616   \exp_args:Nnnx
3617   \begin{structural@feature}{ structure }
3618     { \l__stex_features_structure_name_str }{}
3619     \seq_clear:N \l_tmpa_seq
3620     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3621
3622   }{
3623     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3624     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3625     \str_set:Nx \l_tmpa_str {
3626       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3627       \prop_item:Nn \l_stex_current_module_prop { name }
3628     }
3629     \seq_map_inline:Nn \l_tmpa_seq {
3630       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3631     }
3632     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3633     \exp_args:Nnx
3634     \AddToHookNext { env / mathstructure / after }{
3635       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3636         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3637       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3638     \STEXexport {
3639       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3640       { \prop_item:Nn \l_stex_current_module_prop { origname } }

```

```

3641         {\l_tmpa_str}
3642         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3643         {#2}{\l_tmpa_str}
3644     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3645     % \prop_item:Nn \l_stex_current_module_prop { origname },
3646     % \l_tmpa_str
3647     % }
3648     % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3649     % #2,\l_tmpa_str
3650     % }
3651     % \tl_set:cx { #2 } {
3652     % \stex_invoke_structure:n { \l_tmpa_str }
3653     % }
3654     }
3655
3656 \end{structural@feature}
3657 % \g_stex_last_feature_prop
3658 }

```

\instantiate

```

3659 \seq_new:N \l__stex_features_structure_field_seq
3660 \str_new:N \l__stex_features_structure_field_str
3661 \str_new:N \l__stex_features_structure_def_tl
3662 \prop_new:N \l__stex_features_structure_prop
3663 \NewDocumentCommand \instantiate { m O{} m }{
3664     \stex_smsmode_set_codes:
3665     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3666     \prop_set_eq:Nc \l__stex_features_structure_prop {
3667         c_stex_feature_\l_tmpa_str _prop
3668     }
3669     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3670     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3671         \seq_set_split:Nnn \l_tmpa_seq={}{ ##1 }
3672         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3673             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3674             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3675             {!} \l_tmpa_tl
3676             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3677                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3678                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3679                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3680             }{
3681                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3682                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3683                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3684                 \l_tmpa_tl
3685                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3686                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3687                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3688                 }{
3689                     \tl_clear:N \l_tmpb_tl
3690                 }
3691             }
3692         }{

```

```

3693 \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3694 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3695   \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3696   \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3697   \tl_clear:N \l_tmpa_tl
3698 }{
3699   % TODO throw error
3700 }
3701 }
3702 % \l_tmpa_str: name
3703 % \l_tmpa_tl: definiens
3704 % \l_tmpb_tl: notation
3705 \tl_if_empty:NT \l__stex_features_structure_field_str {
3706   % TODO throw error
3707 }
3708 \str_clear:N \l_tmpb_str
3709
3710 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3711 \seq_map_inline:Nn \l_tmpa_seq {
3712   \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3713   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3714   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3715     \seq_map_break:n {
3716       \str_set:Nn \l_tmpb_str { ###1 }
3717     }
3718   }
3719 }
3720 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3721 \l_tmpb_str
3722
3723 \tl_if_empty:NTF \l_tmpb_tl {
3724   \tl_if_empty:NF \l_tmpa_tl {
3725     \exp_args:Nx \use:n {
3726       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3727     }
3728   }
3729 }{
3730   \tl_if_empty:NTF \l_tmpa_tl {
3731     \exp_args:Nx \use:n {
3732       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3733     }
3734   }
3735 }{
3736   \exp_args:Nx \use:n {
3737     \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3738     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3739   }
3740 }
3741 }
3742 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3743 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3744 % #3/\l__stex_features_structure_field_str
3745 % \par
3746 % \expandafter\present\csname

```

```

3747 %      l_stex_symdecl_
3748 %      \prop_item:Nn \l_stex_current_module_prop {ns} ?
3749 %      \prop_item:Nn \l_stex_current_module_prop {name} ?
3750 %      #3/\l__stex_features_structure_field_str
3751 %      _prop
3752 %      \endcsname
3753 }
3754
3755 \tl_clear:N \l__stex_features_structure_def_tl
3756
3757 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3758 \seq_map_inline:Nn \l_tmpa_seq {
3759   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3760   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3761   \exp_args:Nx \use:n {
3762     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3763
3764     }
3765   }
3766
3767   \prop_if_exist:cF {
3768     l_stex_symdecl_
3769     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3770     \prop_item:Nn \l_stex_current_module_prop {name} ?
3771     #3/\l_tmpa_str
3772     _prop
3773   }{
3774     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3775     \l_tmpb_str
3776     \exp_args:Nx \use:n {
3777       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3778     }
3779   }
3780 }
3781
3782 \symdecl*[type={\STEXsymbol{module-type}}{
3783   \_stex_term_math_oms:nnnn {
3784     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3785     \prop_item:Nn \l__stex_features_structure_prop {name}
3786     }{}{0}{}
3787   }]}{#3}
3788
3789 % TODO: -> sms file
3790
3791 \tl_set:cx{ #3 }{
3792   \stex_invoke_structure:nnn {
3793     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3794     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3795   } {
3796     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3797     \prop_item:Nn \l__stex_features_structure_prop {name}
3798   }
3799 }
3800

```

3801 }

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

\stex_invoke_structure:nnn

```

3802 % #1: URI of the instance
3803 % #2: URI of the instantiated module
3804 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3805   \tl_if_empty:nTF{ #3 }{
3806     \prop_set_eq:Nc \l__stex_features_structure_prop {
3807       c_stex_feature_ #2 _prop
3808     }
3809     \tl_clear:N \l_tmpa_tl
3810     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3811     \seq_map_inline:Nn \l_tmpa_seq {
3812       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3813       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3814       \cs_if_exist:cT {
3815         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3816       }{
3817         \tl_if_empty:NF \l_tmpa_tl {
3818           \tl_put_right:Nn \l_tmpa_tl {,}
3819         }
3820         \tl_put_right:Nx \l_tmpa_tl {
3821           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3822         }
3823       }
3824     }
3825     \exp_args:No \mathstrut \l_tmpa_tl
3826   }{
3827     \stex_invoke_symbol:n{#1/#3}
3828   }
3829 }
```

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

3830 </package>

Chapter 33

STEX -Statements Implementation

```
3831 <*package>
3832
3833 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3834
3835 \protected\def\ignorespacesandpars{
3836   \begingroup\catcode13=10\relax
3837   \@ifnextchar\par{
3838     \endgroup\expandafter\ignorespacesandpars\@gobble
3839   }{
3840     \endgroup
3841   }
3842 }
3843
3844 <@@=stex_statements>
3845
3846   Warnings and error messages
```

\titleemph

```
3846 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3847 \keys_define:nn {stex / definiendum }{
3848   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3849   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3850   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3851 }
3852 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3853   \str_clear:N \l__stex_statements_definiendum_root_str
3854   \tl_clear:N \l__stex_statements_definiendum_post_tl
3855   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3856 \keys_set:nn { stex / definiendum }{ #1 }
3857 }
3858 \NewDocumentCommand \definiendum { 0{ } m m } {
3859   \__stex_statements_definiendum_args:n { #1 }
3860   \stex_get_symbol:n { #2 }
3861   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3862   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3863     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3864       \tl_set:Nn \l_tmpa_tl { #3 }
3865     } {
3866       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3867       \tl_set:Nn \l_tmpa_tl {
3868         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3869       }
3870     }
3871   } {
3872     \tl_set:Nn \l_tmpa_tl { #3 }
3873   }
3874
3875   % TODO root
3876   \rustex_if:TF {
3877     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3878   } {
3879     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3880   }
3881 }
3882 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3883
3884 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3885
3886 \NewDocumentCommand \definame { 0{ } m } {
3887   \__stex_statements_definiendum_args:n { #1 }
3888   % TODO: root
3889   \stex_get_symbol:n { #2 }
3890   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3891   \str_set:Nx \l_tmpa_str {
3892     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3893   }
3894   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3895   \rustex_if:TF {
3896     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3897       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3898     }
3899   } {
3900     \defemph@uri {
3901       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3902     } { \l_stex_get_symbol_uri_str }
3903   }
3904 }
3905 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3906
3907 \NewDocumentCommand \Definame { 0{ } m } {
3908   \_stex_statements_definiendum_args:n { #1 }
3909   \stex_get_symbol:n { #2 }
3910   \str_set:Nx \l_tmpa_str {
3911     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3912   }
3913   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3914   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3915   \rustex_if:TF {
3916     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3917       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3918     }
3919   } {
3920     \defemph@uri {
3921       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3922     } { \l_stex_get_symbol_uri_str }
3923   }
3924 }
3925 \stex_deactivate_macro:Nn \Definame {definition-environments}
3926
3927 \NewDocumentCommand \Symname { 0{ } m }{
3928   \stex_symname_args:n { #1 }
3929   \stex_get_symbol:n { #2 }
3930   \str_set:Nx \l_tmpa_str {
3931     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3932   }
3933   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3934   \let\compemph_uri_prev:\compemph@uri
3935   \let\compemph@uri\symrefemph@uri
3936   \exp_args:NNx \use:nn
3937   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3938     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3939     \l_stex_symname_post_str
3940   ] }
3941   \let\compemph@uri\compemph_uri_prev:
3942 }

```

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

sdefinition

```

3943
3944 \keys_define:nn {stex / sdefinition }{
3945   type      .str_set_x:N = \sdefinitiontype,
3946   id        .str_set_x:N = \sdefinitionid,
3947   title     .tl_set:N    = \sdefinitiontitle
3948 }
3949 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
3950   \str_clear:N \sdefinitiontype
3951   \str_clear:N \sdefinitionid
3952   \tl_clear:N \sdefinitiontitle
3953   \keys_set:nn { stex / sdefinition }{ #1 }
3954 }
3955

```



```

3956 \NewDocumentEnvironment{sdefinition}{0{}}{
3957   \_stex_statements_sdefinition_args:n{ #1 }
3958   \stex_reactivate_macro:N \definiendum
3959   \stex_reactivate_macro:N \definame
3960   \stex_reactivate_macro:N \Definame
3961   \stex_smsmode_set_codes:
3962   \stex_if_smsmode:F {
3963     \exp_args:Nnnx
3964     \begin{stex_annotate_env}{definition}{}
3965     \str_if_empty:NF \sdefinitiontype {
3966       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3967     }
3968   }
3969   \clist_set:No \l_tmpa_clist \sdefinitiontype
3970   \tl_clear:N \l_tmpa_tl
3971   \clist_map_inline:Nn \l_tmpa_clist {
3972     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3973       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3974     }
3975   }
3976   \tl_if_empty:NTF \l_tmpa_tl {
3977     \_stex_statements_sdefinition_start:
3978   }{
3979     \l_tmpa_tl
3980   }
3981   \stex_ref_new_doc_target:n \sdefinitionid
3982 }{
3983   \clist_set:No \l_tmpa_clist \sdefinitiontype
3984   \tl_clear:N \l_tmpa_tl
3985   \clist_map_inline:Nn \l_tmpa_clist {
3986     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3987       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3988     }
3989   }
3990   \tl_if_empty:NTF \l_tmpa_tl {
3991     \_stex_statements_sdefinition_end:
3992   }{
3993     \l_tmpa_tl
3994   }
3995   \stex_if_smsmode:F {
3996     \end{stex_annotate_env}
3997   }
3998 }

```

\stexpatchdefinition

```

3999 \cs_new_protected:Nn \_stex_statements_sdefinition_start: {
4000   \par\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
4001     ~(\sdefinitiontitle)
4002   }~}
4003 }
4004 \cs_new_protected:Nn \_stex_statements_sdefinition_end: {\par\medskip}
4005
4006 \newcommand\stexpatchdefinition[3]{} {
4007   \str_set:Nx \l_tmpa_str{ #1 }

```

```

4008 \str_if_empty:NTF \l_tmpa_str {
4009   \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4010   \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4011 }{
4012   \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4013   \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4014 }
4015 }

```

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

`\inlinedef` inline:

```

4016 \NewDocumentCommand \inlinedef { m } {
4017   \beginingroup
4018   \stex_reactivate_macro:N \definiendum
4019   \stex_reactivate_macro:N \definame
4020   \stex_ref_new_doc_target:n{
4021     #1
4022   }
4023 }

```

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

33.2 Assertions

`sassertion`

```

4024
4025 \keys_define:nn {stex / sassertion }{
4026   type      .str_set_x:N = \sassertiontype,
4027   id        .str_set_x:N = \sassertionid,
4028   title     .tl_set:N     = \sassertiontitle ,
4029   name      .str_set_x:N = \sassertionname
4030 }
4031 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4032   \str_clear:N \sassertiontype
4033   \str_clear:N \sassertionid
4034   \str_clear:N \sassertionname
4035   \tl_clear:N \sassertiontitle
4036   \keys_set:nn { stex / sassertion }{ #1 }
4037 }
4038
4039 %\tl_new:N \g__stex_statements_aftergroup_tl
4040
4041 \NewDocumentEnvironment{sassertion}{0{}}{
4042   \__stex_statements_sassertion_args:n{ #1 }
4043   \stex_smsmode_set_codes:
4044   \stex_if_smsmode:F {
4045     \exp_args:Nnnx
4046     \begin{stex_annotate_env}{assertion}{}
4047     \str_if_empty:NF \sassertiontype {
4048       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4049     }
4050   }

```

```

4051 \clist_set:Nn \l_tmpa_clist \sassertiontype
4052 \tl_clear:N \l_tmpa_tl
4053 \clist_map_inline:Nn \l_tmpa_clist {
4054   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4055     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4056   }
4057 }
4058 \tl_if_empty:NTF \l_tmpa_tl {
4059   \__stex_statements_sassertion_start:
4060 }{
4061   \l_tmpa_tl
4062 }
4063 \stex_ref_new_doc_target:n \sassertionid
4064 }{
4065   \clist_set:Nn \l_tmpa_clist \sassertiontype
4066   \tl_clear:N \l_tmpa_tl
4067   \clist_map_inline:Nn \l_tmpa_clist {
4068     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4069       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4070     }
4071   }
4072   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4073   \tl_if_empty:NTF \l_tmpa_tl {
4074     \__stex_statements_sassertion_end:
4075   }{
4076     \l_tmpa_tl
4077   }
4078   \stex_if_smsmode:F {
4079     \end{stex_annotate_env}
4080   }
4081 }

```

`\stexpatchassertion`

```

4082
4083 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4084   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4085     (\sassertiontitle)
4086   }~}
4087 }
4088 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4089
4090 \newcommand\stexpatchassertion[3] [] {
4091   \str_set:Nx \l_tmpa_str{ #1 }
4092   \str_if_empty:NTF \l_tmpa_str {
4093     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4094     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4095   }{
4096     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4097     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4098   }
4099 }

```

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

`\inlineass` inline:

```

4100 \keys_define:nn {stex / inlineass }{
4101   type      .str_set_x:N = \sassertiontype,
4102   id        .str_set_x:N = \sassertionid,
4103   name      .str_set_x:N = \sassertionname
4104 }
4105 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4106   \str_clear:N \sassertiontype
4107   \str_clear:N \sassertionid
4108   \str_clear:N \sassertionname
4109   \tl_clear:N \sassertiontitle
4110   \keys_set:nn { stex / inlineass }{ #1 }
4111 }
4112 \NewDocumentCommand \inlineass { 0{} m } {
4113   \beginngroup
4114   \__stex_statements_inlineass_args:n{ #1 }
4115   \stex_ref_new_doc_target:n \sassertionid
4116   \stex_annotate:nnn{assertion}{}{
4117     \str_if_empty:NF \sassertiontype {
4118       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4119     }
4120     #1
4121   }
4122   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4123   \endgroup
4124 }

```

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

33.3 Examples

`sexample`

```

4125
4126 \keys_define:nn {stex / sexample }{
4127   type      .str_set_x:N = \exampletype,
4128   id        .str_set_x:N = \sexampleid,
4129   title     .tl_set:N     = \sexampletitle,
4130   for       .clist_set:N  = \sexamplefor,
4131 }
4132 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4133   \str_clear:N \sexampletype
4134   \str_clear:N \sexampleid
4135   \tl_clear:N \sexampletitle
4136   \clist_clear:N \sexamplefor
4137   \keys_set:nn { stex / sexample }{ #1 }
4138 }
4139
4140 \NewDocumentEnvironment{sexample}{0{}}{
4141   \__stex_statements_sexample_args:n{ #1 }
4142   \stex_smsmode_set_codes:
4143   \stex_if_smsmode:F {
4144     \seq_clear:N \l_tmpa_seq
4145     \clist_map_inline:Nn \sexamplefor {

```

```

4146 \str_if_eq:nnF{ ##1 }{}{
4147 \stex_get_symbol:n { ##1 }
4148 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4149 \l_stex_get_symbol_uri_str
4150 }
4151 }
4152 }
4153 \exp_args:Nnnx
4154 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4155 \str_if_empty:NF \sexamplotype {
4156 \stex_annotate_invisible:nnn{type}{\sexamplotype}{}}
4157 }
4158 }
4159 \stex_ref_new_doc_target:n \sexampleid
4160 \clist_set:No \l_tmpa_clist \sexamplotype
4161 \tl_clear:N \l_tmpa_tl
4162 \clist_map_inline:Nn \l_tmpa_clist {
4163 \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4164 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4165 }
4166 }
4167 \tl_if_empty:NTF \l_tmpa_tl {
4168 \__stex_statements_sexample_start:
4169 }{
4170 \l_tmpa_tl
4171 }
4172 }{
4173 \clist_set:No \l_tmpa_clist \sexamplotype
4174 \tl_clear:N \l_tmpa_tl
4175 \clist_map_inline:Nn \l_tmpa_clist {
4176 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4177 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4178 }
4179 }
4180 \tl_if_empty:NTF \l_tmpa_tl {
4181 \__stex_statements_sexample_end:
4182 }{
4183 \l_tmpa_tl
4184 }
4185 \stex_if_smsmode:F {
4186 \end{stex_annotate_env}
4187 }
4188 }

```

\stexpatchexample

```

4189
4190 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4191 \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplotype {
4192 (\sexamplotype)
4193 }~}
4194 }
4195 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4196
4197 \newcommand\stexpatchexample[3] [] {

```

```

4198     \str_set:Nx \l_tmpa_str{ #1 }
4199     \str_if_empty:NTF \l_tmpa_str {
4200         \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4201         \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4202     }{
4203         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4204         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4205     }
4206 }

```

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

`\inlineex` inline:

```

4207 \NewDocumentCommand \inlineex { m } {
4208     \begingroup
4209     \stex_ref_new_doc_target:n{
4210         #1
4211     }
4212 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4213 \keys_define:nn { stex / sparagraph } {
4214     id      .str_set_x:N = \sparagraphid ,
4215     title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4216     type    .str_set_x:N = \sparagraphtype ,
4217     for     .str_set_x:N = \sparagraphfor ,
4218     from    .tl_set_x:N  = \sparagraphfrom ,
4219     start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
4220     name    .str_set:N   = \sparagraphname
4221 }
4222
4223 \cs_new_protected:Nn \stex_sparagraph_args:n {
4224     \tl_clear:N \l_stex_sparagraph_title_tl
4225     \tl_clear:N \sparagraphfrom
4226     \tl_clear:N \l_stex_sparagraph_start_tl
4227     \str_clear:N \sparagraphid
4228     \str_clear:N \sparagraphtype
4229     \str_clear:N \sparagraphfor
4230     \str_clear:N \sparagraphname
4231     \keys_set:nn { stex / sparagraph }{ #1 }
4232 }
4233 \newif\if@in@omtext\@in@omtextfalse
4234
4235 \NewDocumentEnvironment {sparagraph} { 0{} } {
4236     \stex_sparagraph_args:n { #1 }
4237     \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4238         \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4239     }{
4240         \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl

```

```

4241 }
4242 \@in@omtexttrue
4243 \stex_smsmode_set_codes:
4244 \stex_if_smsmode:F {
4245   \exp_args:Nnnx
4246   \begin{stex_annotate_env}{paragraph}{}
4247   \str_if_empty:NF \sparagraphtype {
4248     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4249   }
4250 }
4251 \clist_set:No \l_tmpa_clist \sparagraphtype
4252 \tl_clear:N \l_tmpa_tl
4253 \clist_map_inline:Nn \l_tmpa_clist {
4254   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4255     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4256   }
4257 }
4258 \tl_if_empty:NTF \l_tmpa_tl {
4259   \__stex_statements_sparagraph_start:
4260 }{
4261   \l_tmpa_tl
4262 }
4263 \stex_ref_new_doc_target:n \sparagraphid
4264 \ignorespacesandpars
4265 }{
4266   \clist_set:No \l_tmpa_clist \sparagraphtype
4267   \tl_clear:N \l_tmpa_tl
4268   \clist_map_inline:Nn \l_tmpa_clist {
4269     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4270       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4271     }
4272   }
4273   \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4274   \tl_if_empty:NTF \l_tmpa_tl {
4275     \__stex_statements_sparagraph_end:
4276   }{
4277     \l_tmpa_tl
4278   }
4279   \stex_if_smsmode:F {
4280     \end{stex_annotate_env}
4281   }
4282 }

```

\stexpatchparagraph

```

4283
4284 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4285   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4286     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4287       \titleemph{\l_stex_sparagraph_title_tl}:~
4288     }
4289   }{
4290     \titleemph{\l_stex_sparagraph_start_tl}~
4291   }
4292 }

```

```

4293 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4294
4295 \newcommand\stexpatchparagraph[3] [] {
4296   \str_set:Nx \l_tmpa_str{ #1 }
4297   \str_if_empty:NTF \l_tmpa_str {
4298     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4299     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4300   }{
4301     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4302     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4303   }
4304 }

```

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

symboldoc

```

4305 \NewDocumentEnvironment{symboldoc}{ m }{
4306   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4307   \seq_clear:N \l_tmpb_seq
4308   \seq_map_inline:Nn \l_tmpa_seq {
4309     \str_if_eq:nnF{ ##1 }{}{
4310       \stex_get_symbol:n { ##1 }
4311       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4312         \l_stex_get_symbol_uri_str
4313       }
4314     }
4315   }
4316   \par
4317   \exp_args:Nnnx
4318   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4319 }{
4320   \end{stex_annotate_env}
4321 }
4322 \</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

```
4323 <*package>
4324 <@@=stex_sproof>
4325
4326 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4327
```

34.2 Proofs

We first define some keys for the proof environment.

```
4328 \keys_define:nn { stex / spf } {
4329   id          .str_set:N = \l__stex_sproof_spf_id_str,
4330   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4331   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4332   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4333   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4334   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4335   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4336   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4337   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4338   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4339 }
4340 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4341   \str_clear:N \l__stex_sproof_spf_id_str
4342   \tl_clear:N \l__stex_sproof_spf_display_tl
4343   \tl_clear:N \l__stex_sproof_spf_for_tl
4344   \tl_clear:N \l__stex_sproof_spf_from_tl
4345   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4346   \tl_clear:N \l__stex_sproof_spf_type_tl
4347   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4348 \tl_clear:N \l__stex_sproof_spf_continues_tl
4349 \tl_clear:N \l__stex_sproof_spf_functions_tl
4350 \tl_clear:N \l__stex_sproof_spf_method_tl
4351 \keys_set:nn { stex / spf }{ #1 }
4352 }

```

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

```

4353 \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.

```

4354 \newcount\count_ten
4355 \newenvironment{pst@with@label}[1]{
4356   \edef\pst@label{#1}
4357   \advance\count_ten by 1\relax
4358   \count_ten=1
4359 }{
4360   \advance\count_ten by -1\relax
4361 }

```

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

```

4362 \def\the@pst@label{
4363   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4364 }

```

(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.

```

4365 \keys_define:nn { stex / pstlabel }{
4366   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4367   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4368   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4369 }
4370 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4371 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4372 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4373 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4374 }
4375 \__stex_sproof_pstlabel_args:n {}
4376 \newcommand\setpstlabelstyle[1]{
4377   \__stex_sproof_pstlabel_args:n {#1}
4378 }
4379 \newcommand\setpstlabelstyledefault{%
4380   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4381 }

```

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

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

```

4382 \ExplSyntaxOff
4383 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4384 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4385 \def\pst@make@label@short#1#2{#2}
4386 \def\pst@make@label@empty#1#2{}
4387 \ExplSyntaxOn
4388 \def\pstlabelstyle#1{%
4389   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4390 }%
4391 \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.

```

4392 \def\next@pst@label{%
4393   \global\advance\count\count10 by 1%
4394 }%

```

(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

```

4395 \def\sproof@box{
4396   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4397 }
4398 \def\spf@proofend{\sproof@box}
4399 \def\sproofend{
4400   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4401     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4402   }
4403 }
4404 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4405 \def\spf@proofsketch@kw{Proof Sketch}
4406 \def\spf@proof@kw{Proof}
4407 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4408 \AddToHook{begindocument}{
4409   \ltx@ifpackageloaded{babel}{
4410     \makeatletter
4411     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4412     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4413       \input{sproof-ngerman.ldf}
4414     }
4415     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4416       \input{sproof-finnish.ldf}
4417     }
4418     \clist_if_in:NnT \l_tmpa_clist {french}{
4419       \input{sproof-french.ldf}
4420     }
4421     \clist_if_in:NnT \l_tmpa_clist {russian}{
4422       \input{sproof-russian.ldf}
4423     }
4424     \makeatother
4425   }{}
4426 }

```

`spfsketch`

```

4427 \newcommand\spfsketch[2][]{
4428   \__stex_sproof_spf_args:n{#1}
4429   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4430     \titleemph{
4431       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4432         \spf@proofsketch@kw
4433       }{
4434         \l__stex_sproof_spf_type_tl
4435       }
4436     }:
4437   }
4438   {~#2}
4439   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4440   \sproofend
4441 }

```

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

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

```

4442 \newenvironment{spfeq}[2][]{
4443   \__stex_sproof_spf_args:n{#1}
4444   %\sref@target
4445   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4446     \titleemph{
4447       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4448         \spf@proof@kw
4449       }{

```

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

¹⁵EDNOTE: document above

```

4450         \l__stex_sproof_spf_type_tl
4451     }
4452     }:
4453 }
4454 {~#2}
4455 \begin{displaymath}\begin{array}{rcll}
4456 }{
4457 \end{array}\end{displaymath}
4458 }

```

(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.

```

4459 \newenvironment{spf@proof}[2] []{
4460   \l__stex_sproof_spf_args:n{#1}
4461   %\sref@target
4462   \count_ten=10
4463   \par\noindent
4464   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4465     \titleemph{
4466       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4467         \spf@proof@kw
4468       }{
4469         \l__stex_sproof_spf_type_tl
4470       }
4471     }:
4472   }
4473   {~#2}
4474   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4475   \def\pst@label{}
4476   \newcount\pst@count% initialize the labeling mechanism
4477   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4478   }{
4479     \end{pst@with@label}\end{description}
4480   }
4481   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4482   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4483 \newcommand\spfidea[2] []{
4484   \l__stex_sproof_spf_args:n{#1}
4485   \titleemph{
4486     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4487       \l__stex_sproof_spf_type_tl
4488     }:
4489   }~#2
4490   \sproofend
4491 }

```

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

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

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

EdN:16

```

spfstep 16
4492 \newenvironment{spfstep}[1][]{
4493   \_stex_sproof_spf_args:n{#1}
4494   \@in@omtexttrue
4495   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4496     \item[\the@pst@label]
4497   }
4498   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4499     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4500   }
4501   %\sref@label@id{\pst@label}
4502   \ignorespacesandpars
4503 }{
4504   \next@pst@label\ignorespacesandpars
4505 }

```

sproofcomment

```

4506 \newenvironment{sproofcomment}[1][]{
4507   \_stex_sproof_spf_args:n{#1}
4508   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4509     \item[\the@pst@label]
4510   }
4511 }{
4512   \next@pst@label
4513 }

```

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.

```

4514 \newenvironment{subproof}[2][]{
4515   \_stex_sproof_spf_args:n{#1}
4516   \def\@test{#2}
4517   \ifx\@test\empty\else
4518     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4519       \item[\the@pst@label]
4520     }{#2}
4521   \fi
4522   \begin{pst@with@label}{\pst@label,\number\count_ten}
4523 }{
4524   \end{pst@with@label}\next@pst@label
4525 }

```

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

```

4526 \newenvironment{spfcases}[2][]{
4527   \def\@test{#1}
4528   \ifx\@test\empty
4529     \begin{subproof}[method=by-cases]{#2}

```

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

```

4530 \else
4531   \begin{subproof}[#1,method=by-cases]{#2}
4532 \fi
4533 }{
4534   \end{subproof}
4535 }

```

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

```

4536 \newenvironment{spfcase}[2] [] {
4537   \__stex_sproof_spf_args:n{#1}
4538   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4539     \item[\the@pst@label]
4540   }
4541   \def\@test{#2}
4542   \ifx\@test\@empty
4543   \else
4544     {\titleemph{#2}:~}
4545   \fi
4546   \begin{pst@with@label}{\pst@label,\number\count_ten}
4547 }{
4548   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4549     \sproofend
4550   }
4551   \end{pst@with@label}
4552   \next@pst@label
4553 }

```

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

```

4554 \newcommand\spfcasesketch[3] [] {
4555   \__stex_sproof_spf_args:n{#1}
4556   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4557     \item[\the@pst@label]
4558   }
4559   \def\@test{#2}
4560   \ifx\@test\@empty
4561   \else
4562     {\titleemph{#2}:~}
4563   \fi#3
4564   \next@pst@label
4565 }%

```

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.

```

4566 \keys_define:nn { stex / just }{
4567   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4568   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4569   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4570   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4571 }

```

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

`justification`

4572 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4573 `\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.

4574 `\newcommand\justarg[2] [] {#2}`

4575 `\</package>`

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

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

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

Chapter 35

STEX -Others Implementation

```
4576 <*package>
4577
4578 %%%%%%%%%% others.dtx %%%%%%%%%%
4579
4580 <@@=stex_others>
    Warnings and error messages
4581 % None

\MSC Math subject classifier

4582 \NewDocumentCommand \MSC {m} {
4583 % TODO
4584 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded

4585 \@ifpackageloaded{tikzinput}{
4586 \RequirePackage{stex-tikzinput}
4587 }{}
4588 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4589 \*package>
4590 \@@=stex_modules>
4591
4592 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4593
4594 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4595 \begingroup
4596 \stex_module_setup:nn{
4597   ns=\c_stex_metatheory_ns_str,
4598   meta=NONE
4599 }{Metatheory}
4600 \stex_reactivate_macro:N \symdecl
4601 \stex_reactivate_macro:N \notation
4602 \stex_reactivate_macro:N \symdef
4603 \ExplSyntaxOff
4604 \csname stex_suppress_html:n\endcsname{
4605   % is-a (a:A, a \in A, a is an A, etc.)
4606   \symdecl[args=ai]{isa}
4607   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4608   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4609   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4610
4611   % bind (\forall, \Pi, \lambda etc.)
4612   \symdecl[args=Bi]{bind}
4613   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4614   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4615   \notation[deffun]{bind}{\comp( #1 \comp{ }\; \to\; )}{#1 \comp, #2}
4616
4617   % dummy variable
4618   \symdecl{dummyvar}
4619   \notation[underscore]{dummyvar}{\comp\_}
4620   \notation[dot]{dummyvar}{\comp\cdot}
4621   \notation[dash]{dummyvar}{\comp{\rm --}}
4622
4623   %fromto (function space, Hom-set, implication etc.)
```

```

4624 \symdecl[args=ai]{fromto}
4625 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4626 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4627
4628 % mapto (lambda etc.)
4629 %\symdecl[args=Bi]{mapto}
4630 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4631 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4632 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4633
4634 % function/operator application
4635 \symdecl[args=ia]{apply}
4636 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4637 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4638
4639 % ‘type’ of all collections (sets, classes, types, kinds)
4640 \symdecl{collection}
4641 \notation[U]{collection}{\comp{\mathcal{U}}{}}
4642 \notation[set]{collection}{\comp{\textsf{Set}}{}}
4643
4644 % sequences
4645 \symdecl[args=1]{seqtype}
4646 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4647
4648 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4649 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4650
4651 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4652 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4653 % ^ superceded by \aseqfromto and \livar/\uivar
4654
4655 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4656 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4657 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4658
4659 % letin (‘let’, local definitions, variable substitution)
4660 \symdecl[args=bii]{letin}
4661 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4662 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4663 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4664
4665 % structures
4666 \symdecl*[args=1]{module-type}
4667 \notation{module-type}{\mathtt{MOD} #1}
4668 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4669 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4670
4671 }
4672 \ExplSyntaxOn
4673 \stex_add_to_current_module:n{
4674   \let\nappa\apply
4675   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4676   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4677   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4678 \def\uivar{\csname sequence-index\endcsname[ui]}
4679 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4680 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4681 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4682 }
4683 \__stex_modules_end_module:
4684 \endgroup
4685 \</package>

```

Chapter 37

Tikzinput Implementation

```
4686 <*package>
4687
4688 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4689
4690 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4691 \RequirePackage{l3keys2e}
4692
4693 \keys_define:nn { tikzinput } {
4694   image .bool_set:N = \c_tikzinput_image_bool,
4695   image .default:n = false ,
4696   unknown .code:n = {}
4697 }
4698
4699 \ProcessKeysOptions { tikzinput }
4700
4701 \bool_if:NTF \c_tikzinput_image_bool {
4702   \RequirePackage{graphicx}
4703
4704   \providecommand\usetikzlibrary[]{}
4705   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4706 }{
4707   \RequirePackage{tikz}
4708   \RequirePackage{standalone}
4709
4710   \newcommand \tikzinput [2] [] {
4711     \setkeys{Gin}{#1}
4712     \ifx \Gin@ewidth \Gin@exclamation
4713       \ifx \Gin@eheight \Gin@exclamation
4714         \input { #2 }
4715       \else
4716         \resizebox{!}{ \Gin@eheight }{
4717           \input { #2 }
4718         }
4719       \fi
4720     \else
4721       \ifx \Gin@eheight \Gin@exclamation
4722         \resizebox{ \Gin@ewidth }{!}{
4723           \input { #2 }
```

```

4724     }
4725     \else
4726     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4727     \input { #2 }
4728     }
4729     \fi
4730   \fi
4731 }
4732 }
4733
4734 \newcommand \ctikzinput [2] [] {
4735   \begin{center}
4736     \tikzinput [1] {#2}
4737   \end{center}
4738 }
4739
4740 \@ifpackageloaded{stex}{
4741   \RequirePackage{stex-tikzinput}
4742 }{}
4743
4744 </package>
4745 <*stex>
4746 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4747 \RequirePackage{stex}
4748 \RequirePackage{tikzinput}
4749
4750 \newcommand\mhtikzinput [2] [] {%
4751   \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4752   \stex_in_repository:nn\Gin@mhrepos{
4753     \tikzinput [1]{\mhp\path{##1}{#2}}
4754   }
4755 }
4756 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4757 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

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

```
4758 \*cls)
4759 \@@=document_structure)
4760 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4761 \RequirePackage{l3keys2e,expl-keystr-compat}
```

38.2 Class Options

To initialize the `document-structure` 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`

```
4762 \keys_define:nn{ document-structure / pkg }{
4763   class      .str_set_x:N = \c_document_structure_class_str,
4764   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4765   report     .code:n      = {
4766     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4767     \str_set:Nn \c_document_structure_class_str {report}
4768   },
4769   book       .code:n      = {
4770     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4771     \str_set:Nn \c_document_structure_class_str {book}
4772   },
4773   bookpart   .code:n      = {
4774     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4775     \str_set:Nn \c_document_structure_class_str {book}
4776     \str_set:Nn \c_document_structure_topsect_str {chapter}
4777   },
```

```

4778 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4779 unknown     .code:n      = {
4780   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4781 }
4782 }
4783 \ProcessKeysOptions{ document-structure / pkg }
4784 \str_if_empty:NT \c_document_structure_class_str {
4785   \str_set:Nn \c_document_structure_class_str {article}
4786 }
4787 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4788   {\c_document_structure_class_str}
4789

```

38.3 Beefing up the document environment

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

```

4790 \RequirePackage{document-structure}
4791 \bool_if:NF \c_document_structure_minimal_bool {

```

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_EX_ML.¹⁸

```

4792 \keys_define:nn { document-structure / document }{
4793   id .str_set_x:N = \c_document_structure_document_id_str
4794 }
4795 \let\__document_structure_orig_document=\document
4796 \renewcommand{\document}[1][]{
4797   \keys_set:nn{ document-structure / document }{ #1 }
4798   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4799   \__document_structure_orig_document
4800 }

```

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

```

4801 }
4802 \</cls>

```

38.4 Implementation: document-structure Package

```

4803 \<*package>
4804 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
4805 \RequirePackage{expl-keystr-compat,13keys2e}

```

38.5 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).

¹⁸EDNOTE: faking documentkeys for now. @HANG, please implement


```

4806
4807 \keys_define:nn{ document-structure / pkg }{
4808   class      .str_set_x:N = \c_document_structure_class_str,
4809   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4810   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4811 }
4812 \ProcessKeysOptions{ document-structure / pkg }
4813 \str_if_empty:NT \c_document_structure_class_str {
4814   \str_set:Nn \c_document_structure_class_str {article}
4815 }
4816 \str_if_empty:NT \c_document_structure_topsect_str {
4817   \str_set:Nn \c_document_structure_topsect_str {section}
4818 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded, and set up triggers for other languages

```

4819 \RequirePackage{xspace}
4820 \RequirePackage{comment}
4821 \AddToHook{begindocument}{
4822   \ltx@ifpackageloaded{babel}{
4823     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4824     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4825       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4826     }
4827   }{}
4828 }

```

`\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.

```

4829 \int_new:N \l_document_structure_section_level_int
4830 \str_case:VnF \c_document_structure_topsect_str {
4831   {part}}{
4832     \int_set:Nn \l_document_structure_section_level_int {0}
4833   }
4834   {chapter}}{
4835     \int_set:Nn \l_document_structure_section_level_int {1}
4836   }
4837 }{
4838   \str_case:VnF \c_document_structure_class_str {
4839     {book}}{
4840       \int_set:Nn \l_document_structure_section_level_int {0}
4841     }
4842     {report}}{
4843       \int_set:Nn \l_document_structure_section_level_int {0}
4844     }
4845   }{
4846     \int_set:Nn \l_document_structure_section_level_int {2}
4847   }
4848 }

```

38.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the \LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁹

EdN:19

```
4849 \def\current@section@level{document}%
4850 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4851 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4852 \cs_new_protected:Npn \skipomgroup {
4853   \ifcase\l_document_structure_section_level_int
4854   \or\stepcounter{part}
4855   \or\stepcounter{chapter}
4856   \or\stepcounter{section}
4857   \or\stepcounter{subsection}
4858   \or\stepcounter{subsubsection}
4859   \or\stepcounter{paragraph}
4860   \or\stepcounter{subparagraph}
4861   \fi
4862 }
```

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

`blindomgroup`

```
4863 \newcommand\at@begin@blindomgroup[1]{%
4864 \newenvironment{blindomgroup}
4865 {
4866   \int_incr:N\l_document_structure_section_level_int
4867   \at@begin@blindomgroup\l_document_structure_section_level_int
4868 }{}}
```

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

```
4869 \newcommand\omgroup@nonum[2]{
4870   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4871   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4872 }
```

(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.

```
4873 \newcommand\omgroup@num[2]{
```

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

```

4874 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4875   \@nameuse{#1}{#2}
4876 }{
4877   \cs_if_exist:NTF\rdfmata@sectioning{
4878     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4879   }{
4880     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4881   }
4882 }
4883 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4884 }

```

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

omgroup

```

4885 \keys_define:nn { document-structure / omgroupp }{
4886   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4887   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4888   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4889   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4890   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4891   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4892   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4893   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4894   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4895   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4896 }
4897 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4898   \str_clear:N \l__document_structure_omgroup_id_str
4899   \str_clear:N \l__document_structure_omgroup_date_str
4900   \clist_clear:N \l__document_structure_omgroup_creators_clist
4901   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4902   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4903   \tl_clear:N \l__document_structure_omgroup_type_tl
4904   \tl_clear:N \l__document_structure_omgroup_short_tl
4905   \tl_clear:N \l__document_structure_omgroup_display_tl
4906   \tl_clear:N \l__document_structure_omgroup_intro_tl
4907   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4908   \keys_set:nn { document-structure / omgroupp } { #1 }
4909 }

```

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.

```

4910 \newif\if@mainmatter\@mainmattertrue
4911 \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.

```

4912 \keys_define:nn { document-structure / sectioning }{
4913   name .str_set_x:N = \l__document_structure_sect_name_str ,
4914   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4915   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4916   num .bool_set:N = \l__document_structure_sect_num_bool ,
4917 }

```

```

4918 \cs_new_protected:Nn \__document_structure_sect_args:n {
4919   \str_clear:N \l__document_structure_sect_name_str
4920   \str_clear:N \l__document_structure_sect_ref_str
4921   \bool_set_false:N \l__document_structure_sect_clear_bool
4922   \bool_set_false:N \l__document_structure_sect_num_bool
4923   \keys_set:nn { document-structure / sectioning } { #1 }
4924 }
4925 \newcommand\omdoc@sectioning[3][]{
4926   \__document_structure_sect_args:n {#1}
4927   \let\omdoc@sect@name\l__document_structure_sect_name_str
4928   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4929   \if@mainmatter% numbering not overridden by frontmatter, etc.
4930     \bool_if:NTF \l__document_structure_sect_num_bool {
4931       \omgroup@num{#2}{#3}
4932     }{
4933       \omgroup@nonum{#2}{#3}
4934     }
4935     \def\current@section@level{\omdoc@sect@name}
4936   \else
4937     \omgroup@nonum{#2}{#3}
4938   \fi
4939 }% 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.

```

4940 \newcommand\omgroup@redefine@addtocontents[1]{%
4941   %\edef\__document_structureimport{#1}%
4942   %\@for\@I:=\__document_structureimport\do{%
4943     %\edef\@path{\csname module@\@I @path\endcsname}%
4944     %\@ifundefined{tf@toc}\relax%
4945     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4946   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4947   %\def\addcontentsline##1##2##3{%
4948     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4949   %\else% hyperref.sty not loaded
4950   %\def\addcontentsline##1##2##3{%
4951     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4952   %\fi
4953 }% 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.

```

4954 \int_new:N \l_document_structure_omgroup_level_int
4955 \newenvironment{omgroup}[2][]{% keys, title
4956 {
4957   \__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.

```

4958 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4959   \omgroup@redefine@addtocontents{
4960     %\@ifundefined{module@id}\used@modules%
4961     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4962     }
4963 }

```

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

```

4964 \int_incr:N \l_document_structure_omgroup_level_int
4965 \int_incr:N \l_document_structure_section_level_int
4966 \ifcase\l_document_structure_section_level_int
4967   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4968   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4969   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4970   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4971   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4972   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4973   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4974 \fi
4975 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4976 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4977 }% for customization
4978 {}

```

and finally, we localize the sections

```

4979 \newcommand\omdoc@part@kw{Part}
4980 \newcommand\omdoc@chapter@kw{Chapter}
4981 \newcommand\omdoc@section@kw{Section}
4982 \newcommand\omdoc@subsection@kw{Subsection}
4983 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4984 \newcommand\omdoc@paragraph@kw{paragraph}
4985 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

38.7 Front and Backmatter

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

`\printindex`

```

4986 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

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

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```

4987 \cs_if_exist:NTF\frontmatter{
4988   \let\__document_structure_orig_frontmatter\frontmatter
4989   \let\frontmatter\relax
4990 }{
4991   \tl_set:Nn\__document_structure_orig_frontmatter{
4992     \clearpage
4993     \@mainmatterfalse
4994     \pagenumbering{roman}
4995   }
4996 }

```

```

4997 \cs_if_exist:NTF\backmatter{
4998   \let\__document_structure_orig_backmatter\backmatter
4999   \let\backmatter\relax
5000 }{
5001   \tl_set:Nn\__document_structure_orig_backmatter{
5002     \clearpage
5003     \@mainmatterfalse
5004     \pagenumbering{roman}
5005   }
5006 }

```

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.

```

5007 \newenvironment{frontmatter}{
5008   \__document_structure_orig_frontmatter
5009 }{
5010   \cs_if_exist:NTF\mainmatter{
5011     \mainmatter
5012   }{
5013     \clearpage
5014     \@mainmattertrue
5015     \pagenumbering{arabic}
5016   }
5017 }

```

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

```

5018 \newenvironment{backmatter}{
5019   \__document_structure_orig_backmatter
5020 }{
5021   \cs_if_exist:NTF\mainmatter{
5022     \mainmatter
5023   }{
5024     \clearpage
5025     \@mainmattertrue
5026     \pagenumbering{arabic}
5027   }
5028 }

```

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

```

5029 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5030 \def \c__document_structure_document_str{document}
5031 \newcommand\afterprematurestop{}
5032 \def\prematurestop@endomgroup{
5033   \unless\ifx\@currenvir\c__document_structure_document_str
5034     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
5035     \expandafter\prematurestop@endomgroup
5036   \fi
5037 }

```

```

5038 \providecommand\prematurestop{
5039   \message{Stopping~sTeX~processing~prematurely}
5040   \prematurestop@endomgroup
5041   \afterprematurestop
5042   \end{document}
5043 }

```

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

38.8 Global Variables

`\setSGvar` set a global variable

```

5044 \RequirePackage{etoolbox}
5045 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

`\useSGvar` use a global variable

```

5046 \newrobustcmd\useSGvar[1]{%
5047   \@ifundefined{sTeX@Gvar@#1}
5048   {\PackageError{document-structure}
5049     {The sTeX Global variable #1 is undefined}
5050     {set it with \protect\setSGvar}}
5051   \@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.

```

5052 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5053   \@ifundefined{sTeX@Gvar@#1}
5054   {\PackageError{document-structure}
5055     {The sTeX Global variable #1 is undefined}
5056     {set it with \protect\setSGvar}}
5057   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 39

NotesSlides – Implementation

39.1 Class and Package Options

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

```
5058 \*cls)
5059 \@@=notesslides)
5060 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5061 \RequirePackage{l3keys2e,expl-keystr-compatible}
5062
5063 \keys_define:nn{notesslides / cls}{
5064   class .code:n = {
5065     \PassOptionsToClass{\CurrentOption}{omdoc}
5066     \str_if_eq:nnT{#1}{book}{
5067       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5068     }
5069     \str_if_eq:nnT{#1}{report}{
5070       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5071     }
5072   },
5073   notes .bool_set:N = \c__notesslides_notes_bool ,
5074   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5075   unknown .code:n = {
5076     \PassOptionsToClass{\CurrentOption}{omdoc}
5077     \PassOptionsToClass{\CurrentOption}{beamer}
5078     \PassOptionsToPackage{\CurrentOption}{notesslides}
5079   }
5080 }
5081 \ProcessKeysOptions{ notesslides / cls }
5082 \bool_if:NTF \c__notesslides_notes_bool {
5083   \PassOptionsToPackage{notes=true}{notesslides}
5084 }{
5085   \PassOptionsToPackage{notes=false}{notesslides}
5086 }
5087 \</cls)
```


now we do the same for the notesslides package.

```

5088 <*package>
5089 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5090 \RequirePackage{13keys2e,expl-keystr-compat}
5091
5092 \keys_define:nn{notesslides / pkg}{
5093   topsect      .str_set_x:N = \c_notesslides_topsect_str,
5094   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
5095   notes        .bool_set:N = \c_notesslides_notes_bool ,
5096   slides        .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
5097   sectocframes  .bool_set:N = \c_notesslides_sectocframes_bool ,
5098   frameimages   .bool_set:N = \c_notesslides_frameimages_bool ,
5099   fiboxed       .bool_set:N = \c_notesslides_fiboxed_bool ,
5100   nopproblems   .bool_set:N = \c_notesslides_nopproblems_bool,
5101   unknown       .code:n      = {
5102     \PassOptionsToClass{\CurrentOption}{stex}
5103     \PassOptionsToClass{\CurrentOption}{tikzinput}
5104   }
5105 }
5106 \ProcessKeysOptions{ notesslides / pkg }
5107 \newif\ifnotes
5108 \bool_if:NTF \c_notesslides_notes_bool {
5109   \notesttrue
5110 }{
5111   \notesfalse
5112 }
5113

```

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

```

5114 \str_if_empty:NTF \c_notesslides_topsect_str {
5115   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
5116 }{
5117   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
5118 }
5119 </package>

```

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

```

5120 <*cls>
5121 \bool_if:NTF \c_notesslides_notes_bool {
5122   \LoadClass{document-structure}
5123 }{
5124   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5125   \newcounter{Item}
5126   \newcounter{paragraph}
5127   \newcounter{subparagraph}
5128   \newcounter{Hfootnote}
5129   \RequirePackage{document-structure}
5130 }

```

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

```

5131 \RequirePackage{notesslides}
5132 </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 `notesslides.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.

```

5133 \*package>
5134 \bool_if:NT \c__notesslides_notes_bool {
5135   \RequirePackage{a4wide}
5136   \RequirePackage{marginnote}
5137   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5138   \RequirePackage{mdframed}
5139   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5140   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5141 }
5142 \RequirePackage{stex-tikzinput}
5143 \RequirePackage{etoolbox}
5144 \RequirePackage{amssymb}
5145 \RequirePackage{amsmath}
5146 \RequirePackage{comment}
5147 \RequirePackage{textcomp}
5148 \RequirePackage{url}
5149 \RequirePackage{graphicx}
5150 \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

```

5151 \bool_if:NT \c__notesslides_notes_bool {
5152   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5153 }

```

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

```

5154 \newcounter{slide}
5155 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5156 \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.

```

5157 \bool_if:NTF \c__notesslides_notes_bool {
5158   \renewenvironment{note}{\ignorespaces}{}
5159 }{
5160   \excludcomment{note}
5161 }

```

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.

```
5162 \bool_if:NT \c__notesslides_notes_bool {
5163   \newlength{\slideframewidth}
5164   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5165 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5166   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5167     \bool_set_true:N #1
5168   }{
5169     \bool_set_false:N #1
5170   }
5171 }
5172 \keys_define:nn{notesslides / frame}{
5173   label .str_set_x:N = \l__notesslides_frame_label_str,
5174   allowframebreaks .code:n = {
5175     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5176   },
5177   allowdisplaybreaks .code:n = {
5178     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5179   },
5180   fragile .code:n = {
5181     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5182   },
5183   shrink .code:n = {
5184     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5185   },
5186   squeeze .code:n = {
5187     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5188   },
5189   t .code:n = {
5190     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5191   },
5192 }
5193 \cs_new_protected:Nn \__notesslides_frame_args:n {
5194   \str_clear:N \l__notesslides_frame_label_str
5195   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5196   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5197   \bool_set_true:N \l__notesslides_frame_fragile_bool
5198   \bool_set_true:N \l__notesslides_frame_shrink_bool
5199   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5200   \bool_set_true:N \l__notesslides_frame_t_bool
5201   \keys_set:nn { notesslides / frame }{ #1 }
5202 }
```

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

```
5203 \renewenvironment{frame}[1][]{
5204   \__notesslides_frame_args:n{#1}
5205   \sffamily
5206   \stepcounter{slide}
5207   \def\@currentlabel{\theslide}
5208   \str_if_empty:NF \l__notesslides_frame_label_str {
5209     \label{\l__notesslides_frame_label_str}
```

5210 }

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

```

5211 \def\itemize@level{outer}
5212 \def\itemize@outer{outer}
5213 \def\itemize@inner{inner}
5214 \renewcommand\newpage{\addtocounter{framenum}{1}}
5215 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5216 \renewenvironment{itemize}{
5217   \ifx\itemize@level\itemize@outer
5218     \def\itemize@label{\$ \rhd$}
5219   \fi
5220   \ifx\itemize@level\itemize@inner
5221     \def\itemize@label{\$ \scriptstyle \rhd$}
5222   \fi
5223   \begin{list}
5224     {\itemize@label}
5225     {\setlength{\labelsep}{.3em}
5226      \setlength{\labelwidth}{.5em}
5227      \setlength{\leftmargin}{1.5em}
5228     }
5229   \edef\itemize@level{\itemize@inner}
5230 }{
5231   \end{list}
5232 }

```

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

```

5233 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5234 }{
5235   \medskip\miko@slidelabel\end{mdframed}
5236 }

```

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

`\frametitle`

```

5237 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5238 }

```

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

EdN:21

`\pause` 21

```

5239 \bool_if:NT \c__notesslides_notes_bool {
5240   \newcommand\pause{}
5241 }

```

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

`nparagraph`

```

5242 \bool_if:NTF \c__notesslides_notes_bool {
5243   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}{\end{sparagraph}}
5244 }{
5245   \excludecomment{nparagraph}
5246 }

```

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

```

nomgroup
5247 \bool_if:NTF \c__notesslides_notes_bool {
5248   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5249 }{
5250   \excludecomment{nomgroup}
5251 }

ntheorem
5252 \bool_if:NTF \c__notesslides_notes_bool {
5253   \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5254 }{
5255   \excludecomment{ntheorem}
5256 }

nassertion
5257 \bool_if:NTF \c__notesslides_notes_bool {
5258   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5259 }{
5260   \excludecomment{nassertion}
5261 }

nsproof
5262 \bool_if:NTF \c__notesslides_notes_bool {
5263   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5264 }{
5265   \excludecomment{nsproof}
5266 }

nexample
5267 \bool_if:NTF \c__notesslides_notes_bool {
5268   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
5269 }{
5270   \excludecomment{nexample}
5271 }

nparagraph
5272 \bool_if:NTF \c__notesslides_notes_bool {
5273   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}{\end{sparagraph}}
5274 }{
5275   \excludecomment{nparagraph}
5276 }

\inputref@*skip We customize the hooks for in \inputref.
5277 \def\inputref@preskip{\smallskip}
5278 \def\inputref@postskip{\medskip}

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

\inputref*
5279 \let\orig@inputref\inputref
5280 \def\inputref{\@ifstar\ninputref\orig@inputref}
5281 \newcommand\ninputref[2] [] {
5282   \bool_if:NT \c__notesslides_notes_bool {

```

```

5283     \orig@inputref[#1]{#2}
5284   }
5285 }

```

(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>}`.

```

5286 \newlength{\slidelogoheight}
5287
5288 \bool_if:NTF \c_notesslides_notes_bool {
5289   \setlength{\slidelogoheight}{.4cm}
5290 }{
5291   \setlength{\slidelogoheight}{1cm}
5292 }
5293 \newsavebox{\slidelogo}
5294 \sbox{\slidelogo}{\sTeX}
5295 \newrobustcmd{\setslidelogo}[1]{
5296   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5297 }

```

(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.

```

5298 \def\source{Michael Kohlhase}% customize locally
5299 \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.

```

5300 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5301 \newsavebox{\cclogo}
5302 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5303 \newif\ifcchref\cchreffalse
5304 \AtBeginDocument{
5305   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5306 }
5307 \def\licensing{
5308   \ifcchref
5309     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5310   \else
5311     {\usebox{\cclogo}}
5312   \fi
5313 }

```

```

5314 \newrobustcmd{\setlicensing}[2][]{
5315   \def\@url{#1}
5316   \sbox{\cclogo}{\includegraphics[height=\slideologoheight]{#2}}
5317   \ifx\@url\@empty
5318     \def\licensing{\usebox{\cclogo}}
5319   \else
5320     \def\licensing{
5321       \ifcchref
5322         \href{#1}{\usebox{\cclogo}}
5323       \else
5324         {\usebox{\cclogo}}
5325       \fi
5326     }
5327   \fi
5328 }

```

(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

```

5329 \newrobustcmd\miko@slidelabel{
5330   \vbox to \slideologoheight{
5331     \vss\hbox to \slidewidth
5332     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slideologo}}
5333   }
5334 }

```

(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.

```

5335 \def\Gin@mhrepos{}
5336 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5337 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5338 \newrobustcmd\frameimage[2][]{
5339   \stepcounter{slide}
5340   \bool_if:NT \c__notesslides_frameimages_bool {
5341     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5342     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5343     \begin{center}
5344       \bool_if:NTF \c__notesslides_fiboxed_bool {
5345         \fbox{
5346           \ifx\Gin@ewidth\@empty
5347             \ifx\Gin@mhrepos\@empty
5348               \mhgraphics[width=\slidewidth,#1]{#2}
5349             \else
5350               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5351             \fi
5352           \else% Gin@ewidth empty
5353             \ifx\Gin@mhrepos\@empty
5354               \mhgraphics[#1]{#2}

```

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

```

5355         \else
5356             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5357         \fi
5358     \fi% Gin@ewidth empty
5359 }
5360 }{
5361     \ifx\Gin@ewidth\@empty
5362         \ifx\Gin@mhrepos\@empty
5363             \mhgraphics[width=\slidewidth,#1]{#2}
5364         \else
5365             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5366         \fi
5367         \ifx\Gin@mhrepos\@empty
5368             \mhgraphics[#1]{#2}
5369         \else
5370             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5371         \fi
5372     \fi% Gin@ewidth empty
5373 }
5374 \end{center}
5375 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
5376 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5377 }
5378 } % 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.

```

5379 \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

```

5380 \AddToHook{begindocument}{
5381     \definecolor{green}{rgb}{0,.5,0}
5382     \definecolor{purple}{cmyk}{.3,1,0,.17}
5383 }

```

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.

```

5384 % \def\STpresent#1{\textcolor{blue}{#1}}
5385 \def\defemph#1{\textcolor{magenta}{#1}}
5386 \def\symrefemph#1{\textcolor{cyan}{#1}}
5387 \def\compemph#1{\textcolor{blue}{#1}}
5388 \def\titleemph#1{\textcolor{blue}{#1}}
5389 \def\__omtextlec#1{\textcolor{green}{#1}}

```

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

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

```

5390 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5391 \def\smalltextwarning{
5392   \pgfuseimage{miko@small@dbend}
5393   \xspace
5394 }
5395 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
5396 \newrobustcmd\textwarning{
5397   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5398   \xspace
5399 }
5400 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5401 \newrobustcmd\bigtextwarning{
5402   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5403   \xspace
5404 }

```

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

```

5405 \newrobustcmd\putgraphicsat[3]{
5406   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5407 }
5408 \newrobustcmd\putat[2]{
5409   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5410 }

```

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`.

```

5411 \bool_if:NT \c__notesslides_sectocframes_bool {
5412   \str_if_eq:VnTF \__notesslidestopsect{part}{
5413     \newcounter{chapter}\counterwithin*{section}{chapter}
5414   }{
5415     \str_if_eq:VnTF \__notesslidestopsect{chapter}{
5416       \newcounter{chapter}\counterwithin*{section}{chapter}
5417     }
5418   }
5419 }

```

`\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
5420 \def\part@prefix{}
5421 \@ifpackageloaded{document-structure}{}{
5422   \str_case:VnF \__notesslidestopsect {
5423     {part}{
5424       \int_set:Nn \l_document_structure_section_level_int {0}
5425       \def\thesection{\arabic{chapter}.\arabic{section}}
5426       \def\part@prefix{\arabic{chapter}.}
5427     }

```

```

5428 {chapter}{
5429   \int_set:Nn \l_document_structure_section_level_int {1}
5430   \def\thesection{\arabic{chapter}.\arabic{section}}
5431   \def\part@prefix{\arabic{chapter}.}
5432 }
5433 }{
5434   \int_set:Nn \l_document_structure_section_level_int {2}
5435   \def\part@prefix{}
5436 }
5437 }
5438
5439 \bool_if:NF \c__notesslides_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

```

5440 \renewenvironment{omgroup}[2][]{
5441   \_document_structure_omgroup_args:n { #1 }
5442   \int_incr:N \l_document_structure_omgroup_level_int
5443   \int_incr:N \l_document_structure_section_level_int
5444   \bool_if:NT \c__notesslides_sectocframes_bool {
5445     \stepcounter{slide}
5446     \begin{frame}[noframenumbering]
5447     \vfill\Large\centering
5448     \red{
5449       \ifcase\l_document_structure_section_level_int\or
5450         \stepcounter{part}
5451         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
5452         \def\currentsectionlevel{\omdoc@part@kw}
5453       \or
5454         \stepcounter{chapter}
5455         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5456         \def\currentsectionlevel{\omdoc@chapter@kw}
5457       \or
5458         \stepcounter{section}
5459         \def\_notesslideslabel{\part@prefix\arabic{section}}
5460         \def\currentsectionlevel{\omdoc@section@kw}
5461       \or
5462         \stepcounter{subsection}
5463         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5464         \def\currentsectionlevel{\omdoc@subsection@kw}
5465       \or
5466         \stepcounter{subsubsection}
5467         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5468         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5469       \or
5470         \stepcounter{paragraph}
5471         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5472         \def\currentsectionlevel{\omdoc@paragraph@kw}
5473       \else
5474         \def\_notesslideslabel{}
5475         \def\currentsectionlevel{\omdoc@paragraph@kw}

```

```

5476         \fi% end ifcase
5477         \_notesslideslabel%\sref@label@id\_notesslideslabel
5478         \quad #2%
5479     }%
5480     \vfill%
5481     \end{frame}%
5482 }
5483 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str%
5484 }{}
5485 }

```

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

```

5486 \def\inserttheorembodyfont{\normalfont}
5487 %\bool_if:NF \c__notesslides_notes_bool {
5488 % \defbeamertemplate{theorem begin}{miko}
5489 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5490 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5491 % \inserttheorem punctuation\inserttheorembodyfont\xspace}
5492 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5493 % \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.

```

5494 % \expandafter\def\csname Parent2\endcsname{}
5495 %}
5496
5497 \AddToHook{begindocument}{% this does not work for some reason
5498 \setbeamertemplate{theorems}[ams style]
5499 }
5500 \bool_if:NT \c__notesslides_notes_bool {
5501 \renewenvironment{columns}[1][{}]{%
5502 \par\noindent%
5503 \begin{minipage}%
5504 \slidewidth\centering\leavevmode%
5505 }{%
5506 \end{minipage}\par\noindent%
5507 }%
5508 \newsavebox\columnbox%
5509 \renewenvironment<>{column}[2][{}]{%
5510 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5511 }{%
5512 \end{minipage}\end{lrbox}\usebox\columnbox%
5513 }%
5514 }
5515 \bool_if:NTF \c__notesslides_noproblems_bool {
5516 \newenvironment{problems}{}{}
5517 }{
5518 \excludecomment{problems}
5519 }

```

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.

```

5520 \gdef\printexcursions{}
5521 \newcommand\excursionref[2]{% label, text
5522   \bool_if:NT \c__notesslides_notes_bool {
5523     \begin{sparagraph}[title=Excursion]
5524       #2 \sref[fallback=the appendix]{#1}.
5525     \end{sparagraph}
5526   }
5527 }
5528 \newcommand\activate@excursion[2][]{
5529   \gappto\printexcursions{\inputref{#1}{#2}}
5530 }
5531 \newcommand\excursion[4][]{% repos, label, path, text
5532   \bool_if:NT \c__notesslides_notes_bool {
5533     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5534   }
5535 }

```

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

`\excursiongroup`

```

5536 \keys_define:nn{notesslides / excursiongroup }{
5537   id          .str_set_x:N = \l__notesslides_excursion_id_str,
5538   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
5539   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5540 }
5541 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5542   \tl_clear:N \l__notesslides_excursion_intro_tl
5543   \str_clear:N \l__notesslides_excursion_id_str
5544   \str_clear:N \l__notesslides_excursion_mhrepos_str
5545   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5546 }
5547 \newcommand\excursiongroup[1][]{
5548   \__notesslides_excursion_args:n{ #1 }
5549   \ifdefempty\printexcursions{}% only if there are excursions
5550   {\begin{note}
5551     \begin{omgroup}[#1]{Excursions}%
5552     \ifdefempty\l__notesslides_excursion_intro_tl{\{
5553       \inputref[\l__notesslides_excursion_mhrepos_str]{
5554         \l__notesslides_excursion_intro_tl
5555       }
5556     }
5557     \printexcursions%
5558     \end{omgroup}
5559     \end{note}}
5560 }
5561 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5562 \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.

```
5563 <*package>
5564 <@@=problems>
5565 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5566 \RequirePackage{l3keys2e,expl-keystr-compatible}
5567
5568 \keys_define:nn { problem / pkg }{
5569   notes      .default:n    = { true },
5570   notes      .bool_set:N   = \c__problems_notes_bool,
5571   gnotes     .default:n    = { true },
5572   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5573   hints      .default:n    = { true },
5574   hints      .bool_set:N   = \c__problems_hints_bool,
5575   solutions  .default:n    = { true },
5576   solutions  .bool_set:N   = \c__problems_solutions_bool,
5577   pts        .default:n    = { true },
5578   pts        .bool_set:N   = \c__problems_pts_bool,
5579   min        .default:n    = { true },
5580   min        .bool_set:N   = \c__problems_min_bool,
5581   boxed      .default:n    = { true },
5582   boxed      .bool_set:N   = \c__problems_boxed_bool,
5583   unknown    .code:n       = {}
5584 }
5585 \newif\ifsolutions
5586
5587 \ProcessKeysOptions{ problem / pkg }
5588 \bool_if:NTF \c__problems_solutions_bool {
5589   \solutionstrue
5590 }{
5591   \solutionsfalse
5592 }
```

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

```
5593 \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.

```
5594 \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.

```
5595 \def\prob@problem@kw{Problem}
5596 \def\prob@solution@kw{Solution}
5597 \def\prob@hint@kw{Hint}
5598 \def\prob@note@kw{Note}
5599 \def\prob@gnote@kw{Grading}
5600 \def\prob@pt@kw{pt}
5601 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5602 \AddToHook{begindocument}{
5603   \ltx@ifpackageloaded{babel}{
5604     \makeatletter
5605     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5606     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5607       \input{problem-ngerman.ldf}
5608     }
5609     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5610       \input{problem-finnish.ldf}
5611     }
5612     \clist_if_in:NnT \l_tmpa_clist {french}{
5613       \input{problem-french.ldf}
5614     }
5615     \clist_if_in:NnT \l_tmpa_clist {russian}{
5616       \input{problem-russian.ldf}
5617     }
5618     \makeatother
5619   }{ }
5620 }
```

40.2 Problems and Solutions

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

```
5621 \keys_define:nn{ problem / problem }{
5622   id      .str_set:x:N = \l__problems_prob_id_str,
5623   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5624   min     .tl_set:N    = \l__problems_prob_min_tl,
5625   title   .tl_set:N    = \l__problems_prob_title_tl,
5626   type    .tl_set:N    = \l__problems_prob_type_tl,
5627   refnum  .int_set:N    = \l__problems_prob_refnum_int
5628 }
5629 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5630 \str_clear:N \l__problems_prob_id_str
5631 \tl_clear:N \l__problems_prob_pts_tl
5632 \tl_clear:N \l__problems_prob_min_tl
5633 \tl_clear:N \l__problems_prob_title_tl
5634 \tl_clear:N \l__problems_prob_type_tl
5635 \int_zero_new:N \l__problems_prob_refnum_int
5636 \keys_set:nn { problem / problem }{ #1 }
5637 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5638   \let\l__problems_prob_refnum_int\undefined
5639 }
5640 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5641 \newcounter{problem}
5642 \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.

```

5643 \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

```

5644 \newcommand\prob@number{
5645   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5646     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5647   }{
5648     \int_if_exist:NTF \l__problems_prob_refnum_int {
5649       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5650     }{
5651       \prob@label\theproblem
5652     }
5653   }
5654 }

```

(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.

```

5655 \newcommand\prob@title[3]{%
5656   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5657     #2 \l__problems_inclprob_title_tl #3
5658   }{
5659     \tl_if_exist:NTF \l__problems_prob_title_tl {
5660       #2 \l__problems_prob_title_tl #3
5661     }{
5662       #1
5663     }
5664   }
5665 }

```

(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.

```
5666 \def\prob@heading{
5667   {\prob@problem@kw}~\prob@number\prob@title{~}{~}{~}\strut}
5668   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5669 }
```

(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.

sproblem

```
5670 \newenvironment{sproblem}[1][{}]{
5671   \__problems_prob_args:n{#1}%\sref@target%
5672   \@in@omtexttrue% we are in a statement (for inline definitions)
5673   \stepcounter{problem}\record@problem
5674   \def\current@section@level{\prob@problem@kw}
5675   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5676     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5677   }{
5678     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5679   }
5680   \str_if_exist:NTF \l__problems_inclprob_id_str {
5681     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5682   }{
5683     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5684   }
5685
5686
5687   \clist_set:No \l_tmpa_clist \sproblemtype
5688   \tl_clear:N \l_tmpa_tl
5689   \clist_map_inline:Nn \l_tmpa_clist {
5690     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5691       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5692     }
5693   }
5694   \tl_if_empty:NTF \l_tmpa_tl {
5695     \__problems_sproblem_start:
5696   }{
5697     \l_tmpa_tl
5698   }
5699   \stex_ref_new_doc_target:n \sproblemid
5700 }{
5701   \clist_set:No \l_tmpa_clist \sproblemtype
5702   \tl_clear:N \l_tmpa_tl
5703   \clist_map_inline:Nn \l_tmpa_clist {
5704     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5705       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5706     }
5707   }
```



```

5707 }
5708 \tl_if_empty:NTF \l_tmpa_tl {
5709   \__problems_sproblem_end:
5710 }{
5711   \l_tmpa_tl
5712 }
5713
5714
5715 \smallskip
5716 }
5717
5718
5719 \cs_new_protected:Nn \__problems_sproblem_start: {
5720   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5721 }
5722 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5723
5724 \newcommand\stexpatchproblem[3][] {
5725   \str_set:Nx \l_tmpa_str{ #1 }
5726   \str_if_empty:NTF \l_tmpa_str {
5727     \tl_set:Nn \__problems_sproblem_start: { #2 }
5728     \tl_set:Nn \__problems_sproblem_end: { #3 }
5729   }{
5730     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5731     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5732   }
5733 }
5734
5735
5736 \bool_if:NT \c__problems_boxed_bool {
5737   \surroundwithmdframed{problem}
5738 }

```

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

```

5739 \def\record@problem{
5740   \protected@write\@auxout{}
5741   {
5742     \string\@problem{\prob@number}
5743     {
5744       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5745         \l__problems_inclprob_pts_tl
5746       }{
5747         \l__problems_prob_pts_tl
5748       }
5749     }%
5750     {
5751       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5752         \l__problems_inclprob_min_tl
5753       }{
5754         \l__problems_prob_min_tl
5755       }
5756     }
5757   }
5758 }

```

(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).

```
5759 \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.

```
5760 \keys_define:nn { problem / solution }{
5761   id                .str_set_x:N = \l__problems_solution_id_str ,
5762   for               .tl_set:N   = \l__problems_solution_for_tl ,
5763   height           .dim_set:N   = \l__problems_solution_height_dim ,
5764   creators         .clist_set:N = \l__problems_solution_creators_clist ,
5765   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
5766   srccite          .tl_set:N    = \l__problems_solution_srccite_tl
5767 }
5768 \cs_new_protected:Nn \__problems_solution_args:n {
5769   \str_clear:N \l__problems_solution_id_str
5770   \tl_clear:N \l__problems_solution_for_tl
5771   \tl_clear:N \l__problems_solution_srccite_tl
5772   \clist_clear:N \l__problems_solution_creators_clist
5773   \clist_clear:N \l__problems_solution_contributors_clist
5774   \dim_zero:N \l__problems_solution_height_dim
5775   \keys_set:nn { problem / solution }{ #1 }
5776 }
```

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

```
5777 \newcommand\@startsolution[1][{}]{
5778   \__problems_solution_args:n { #1 }
5779   \@in@omtexttrue% we are in a statement.
5780   \bool_if:NF \c__problems_boxed_bool { \hrule }
5781   \smallskip\noindent
5782   {\textbf\prob@solution@kw : \enspace}
5783   \begin{small}
5784   \def\current@section@level{\prob@solution@kw}
5785   \ignorespacesandpars
5786 }
```

\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.

```
5787 \newcommand\startsolutions{
5788   \specialcomment{solution}{\@startsolution}{
5789     \bool_if:NF \c__problems_boxed_bool {
5790       \hrule\medskip
5791     }
5792     \end{small}%
5793   }
5794   \bool_if:NT \c__problems_boxed_bool {
5795     \surroundwithmdframed{solution}
5796   }
5797 }
```

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

\stopsolutions

```
5798 \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.

```
5799 \ifsolutions
5800 \startsolutions
5801 \else
5802 \stopsolutions
5803 \fi
```

exnote

```
5804 \bool_if:NTF \c__problems_notes_bool {
5805 \newenvironment{exnote}[1][]{
5806 \par\smallskip\hrule\smallskip
5807 \noindent\textbf{\prob@note@kw : }\small
5808 }{
5809 \smallskip\hrule
5810 }
5811 }{
5812 \excludecomment{exnote}
5813 }
```

hint

```
5814 \bool_if:NTF \c__problems_notes_bool {
5815 \newenvironment{hint}[1][]{
5816 \par\smallskip\hrule\smallskip
5817 \noindent\textbf{\prob@hint@kw :~ }\small
5818 }{
5819 \smallskip\hrule
5820 }
5821 \newenvironment{exhint}[1][]{
5822 \par\smallskip\hrule\smallskip
5823 \noindent\textbf{\prob@hint@kw :~ }\small
5824 }{
5825 \smallskip\hrule
5826 }
5827 }{
5828 \excludecomment{hint}
5829 \excludecomment{exhint}
5830 }
```

gnote

```
5831 \bool_if:NTF \c__problems_notes_bool {
5832 \newenvironment{gnote}[1][]{
5833 \par\smallskip\hrule\smallskip
5834 \noindent\textbf{\prob@gnote@kw : }\small
5835 }{
5836 \smallskip\hrule
5837 }
5838 }{
5839 \excludecomment{gnote}
5840 }
```

40.3 Multiple Choice Blocks

```

5841 \newenvironment{mcb}{
5842   \begin{enumerate}
5843 }{
5844   \end{enumerate}
5845 }

```

we define the keys for the mcc macro

```

5846 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5847   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5848     \bool_set_true:N #1
5849   }{
5850     \bool_set_false:N #1
5851   }
5852 }
5853 \keys_define:nn { problem / mcc }{
5854   id          .str_set:x:N = \l__problems_mcc_id_str ,
5855   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5856   T           .default:n   = { true } ,
5857   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5858   F           .default:n   = { true } ,
5859   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5860   Ttext       .code:n       = {
5861     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5862   } ,
5863   Ftext       .code:n       = {
5864     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5865   }
5866 }
5867 \cs_new_protected:Nn \l__problems_mcc_args:n {
5868   \str_clear:N \l__problems_mcc_id_str
5869   \tl_clear:N \l__problems_mcc_feedback_tl
5870   \bool_set_true:N \l__problems_mcc_t_bool
5871   \bool_set_true:N \l__problems_mcc_f_bool
5872   \bool_set_true:N \l__problems_mcc_Ttext_bool
5873   \bool_set_false:N \l__problems_mcc_Ftext_bool
5874   \keys_set:nn { problem / mcc }{ #1 }
5875 }

```

\mcc

```

5876 \newcommand\mcc[2][] {
5877   \l__problems_mcc_args:n{ #1 }
5878   \item #2
5879   \ifsolutions
5880     \\\
5881     \bool_if:NT \l__problems_mcc_t_bool {
5882       % TODO!
5883       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
5884     }
5885     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

5886         % TODO!
5887         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5888     }
5889     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5890         !
5891     }{
5892         \l__problems_mcc_feedback_tl
5893     }
5894     \fi
5895 } %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.

```

5896
5897 \keys_define:nn{ problem / inclproblem }{
5898   id      .str_set:N = \l__problems_inclprob_id_str,
5899   pts     .tl_set:N   = \l__problems_inclprob_pts_tl,
5900   min     .tl_set:N   = \l__problems_inclprob_min_tl,
5901   title   .tl_set:N   = \l__problems_inclprob_title_tl,
5902   refnum  .int_set:N   = \l__problems_inclprob_refnum_int,
5903   type    .tl_set:N   = \l__problems_inclprob_type_tl,
5904   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
5905 }
5906 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5907   \str_clear:N \l__problems_prob_id_str
5908   \tl_clear:N \l__problems_inclprob_pts_tl
5909   \tl_clear:N \l__problems_inclprob_min_tl
5910   \tl_clear:N \l__problems_inclprob_title_tl
5911   \tl_clear:N \l__problems_inclprob_type_tl
5912   \int_zero_new:N \l__problems_inclprob_refnum_int
5913   \str_clear:N \l__problems_inclprob_mhrepos_str
5914   \keys_set:nn { problem / inclproblem }{ #1 }
5915   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5916     \let\l__problems_inclprob_pts_tl\undefined
5917   }
5918   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5919     \let\l__problems_inclprob_min_tl\undefined
5920   }
5921   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5922     \let\l__problems_inclprob_title_tl\undefined
5923   }
5924   \tl_if_empty:NT \l__problems_inclprob_type_tl {
5925     \let\l__problems_inclprob_type_tl\undefined
5926   }
5927   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5928     \let\l__problems_inclprob_refnum_int\undefined
5929   }
5930 }

```

```

5931
5932 \cs_new_protected:Nn \__problems_inclprob_clear: {
5933   \let\l__problems_inclprob_id_str\undefined
5934   \let\l__problems_inclprob_pts_tl\undefined
5935   \let\l__problems_inclprob_min_tl\undefined
5936   \let\l__problems_inclprob_title_tl\undefined
5937   \let\l__problems_inclprob_type_tl\undefined
5938   \let\l__problems_inclprob_refnum_int\undefined
5939   \let\l__problems_inclprob_mhrepos_str\undefined
5940 }
5941 \__problems_inclprob_clear:
5942
5943 \newcommand\includeproblem[2][ ]{
5944   \__problems_inclprob_args:n{ #1 }
5945   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5946     \input{#2}
5947   }{
5948     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5949       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5950     }
5951   }
5952   \__problems_inclprob_clear:
5953 }

```

(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.

```

5954 \AddToHook{enddocument}{
5955   \bool_if:NT \c__problems_pts_bool {
5956     \message{Total:~\arabic{pts}~points}
5957   }
5958   \bool_if:NT \c__problems_min_bool {
5959     \message{Total:~\arabic{min}~minutes}
5960   }
5961 }

```

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

```

5962 \def\pts#1{
5963   \bool_if:NT \c__problems_pts_bool {
5964     \marginpar{#1~\prob@pt@kw}
5965   }
5966 }
5967 \def\min#1{
5968   \bool_if:NT \c__problems_min_bool {
5969     \marginpar{#1~\prob@min@kw}
5970   }
5971 }

```

`\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.

```

5972 \newcounter{pts}
5973 \def\show@pts{
5974   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5975     \bool_if:NT \c__problems_pts_bool {
5976       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5977       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
5978     }
5979   }{
5980     \tl_if_exist:NT \l__problems_prob_pts_tl {
5981       \bool_if:NT \c__problems_pts_bool {
5982         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5983         \addtocounter{pts}{\l__problems_prob_pts_tl}
5984       }
5985     }
5986   }
5987 }

```

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

`\show@min`

```

5988 \newcounter{min}
5989 \def\show@min{
5990   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5991     \bool_if:NT \c__problems_min_bool {
5992       \marginpar{\l__problems_inclprob_min_tl;min}
5993       \addtocounter{min}{\l__problems_inclprob_min_tl}
5994     }
5995   }{
5996     \tl_if_exist:NT \l__problems_prob_min_tl {
5997       \bool_if:NT \c__problems_min_bool {
5998         \marginpar{\l__problems_prob_min_tl;min}
5999         \addtocounter{min}{\l__problems_prob_min_tl}
6000       }
6001     }
6002   }
6003 }
6004 \</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.

```
6005 <@@=hwexam>
6006 <*cls>
6007 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6008 \RequirePackage{l3keys2e,expl-keystr-compatible}
6009 \DeclareOption*{
6010   \PassOptionsToClass{\CurrentOption}{document-structure}
6011   \PassOptionsToPackage{\CurrentOption}{stex}
6012   \PassOptionsToPackage{\CurrentOption}{hwexam}
6013   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6014 }
6015 \ProcessOptions
```

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

```
6016 \LoadClass{document-structure}
6017 \RequirePackage{stex}
6018 \RequirePackage{hwexam}
6019 \RequirePackage{tikzinput}
6020 \RequirePackage{graphicx}
6021 \RequirePackage{a4wide}
6022 \RequirePackage{amssymb}
6023 \RequirePackage{amstext}
6024 \RequirePackage{amsmath}
```

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


```

6025 \newcommand\assig@default@type{\hwexam@assignment@kw}
6026 \def\document@hwexamtype{\assig@default@type}
6027 <@@=document_structure>
6028 \keys_define:nn { document-structure / document }{
6029 id .str_set_x:N = \c_document_structure_document_id_str,
6030 hwexamtype .tl_set:N = \document@hwexamtype
6031 }
6032 <@@=hwexam>
6033 </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.

```
6034 \*package>
6035 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6036 \RequirePackage{l3keys2e,expl-keystr-compat}
6037
6038 \newif\iftest\testfalse
6039 \DeclareOption{test}{\testtrue}
6040 \newif\ifmultiple\multiplefalse
6041 \DeclareOption{multiple}{\multipletrue}
6042 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6043 \ProcessOptions
```

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

```
6044 \RequirePackage{keyval}[1997/11/10]
6045 \RequirePackage{problem}
```

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

```
6046 \newcommand\hwexam@assignment@kw{Assignment}
6047 \newcommand\hwexam@given@kw{Given}
6048 \newcommand\hwexam@due@kw{Due}
6049 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6050 blank~for~extra~space}
6051 \def\hwexam@minutes@kw{minutes}
6052 \newcommand\correction@probs@kw{prob.}
6053 \newcommand\correction@pts@kw{total}
6054 \newcommand\correction@reached@kw{reached}
6055 \newcommand\correction@sum@kw{Sum}
6056 \newcommand\correction@grade@kw{grade}
6057 \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

```

6058 \AddToHook{begindocument}{
6059 \ltx@ifpackageloaded{babel}{
6060 \makeatletter
6061 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6062 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6063 \input{hwexam-ngerman.ldf}
6064 }
6065 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6066 \input{hwexam-finnish.ldf}
6067 }
6068 \clist_if_in:NnT \l_tmpa_clist {french}{
6069 \input{hwexam-french.ldf}
6070 }
6071 \clist_if_in:NnT \l_tmpa_clist {russian}{
6072 \input{hwexam-russian.ldf}
6073 }
6074 \makeatother
6075 }{}
6076 }
6077

```

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.

```

6078 \newcounter{assignment}
6079 \numberproblemsin{assignment}
6080 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6081 \keys_define:nn { hwexam / assignment } {
6082 id .str_set:N = \l__hwexam_assign_id_str,
6083 number .int_set:N = \l__hwexam_assign_number_int,
6084 title .tl_set:N = \l__hwexam_assign_title_tl,
6085 type .tl_set:N = \l__hwexam_assign_type_tl,
6086 given .tl_set:N = \l__hwexam_assign_given_tl,
6087 due .tl_set:N = \l__hwexam_assign_due_tl,
6088 loadmodules .code:n = {
6089 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6090 }
6091 }
6092 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6093 \str_clear:N \l__hwexam_assign_id_str
6094 \int_set:Nn \l__hwexam_assign_number_int {-1}
6095 \tl_clear:N \l__hwexam_assign_title_tl
6096 \tl_clear:N \l__hwexam_assign_type_tl
6097 \tl_clear:N \l__hwexam_assign_given_tl
6098 \tl_clear:N \l__hwexam_assign_due_tl
6099 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6100 \keys_set:nn { hwexam / assignment }{ #1 }
6101 }

```

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.

```

6102 \newcommand\given@due[2]{
6103 \bool_lazy_all:nF {
6104 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6105 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6106 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6107 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6108 }{ #1 }
6109
6110 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6111 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6112 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6113 }
6114 }{
6115 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6116 }
6117
6118 \bool_lazy_or:nnF {
6119 \bool_lazy_and_p:nn {
6120 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6121 }{
6122 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6123 }
6124 }{
6125 \bool_lazy_and_p:nn {
6126 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6127 }{
6128 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6129 }
6130 }{ ,~ }
6131
6132 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6133 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6134 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6135 }
6136 }{
6137 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6138 }
6139
6140 \bool_lazy_all:nF {
6141 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6142 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6143 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6144 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6145 }{ #2 }
6146 }

```

`\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.

```

6147 \newcommand\assignment@title[3]{
6148 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6149 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6150 #1
6151 }{
6152 #2\l__hwexam_assign_title_tl#3
6153 }
6154 }{
6155 #2\l__hwexam_inclasssign_title_tl#3
6156 }
6157 }

```

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

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

```

6158 \newcommand\assignment@number{
6159 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6160 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6161 \arabic{assignment}
6162 } {
6163 \int_use:N \l__hwexam_assign_number_int
6164 }
6165 }{
6166 \int_use:N \l__hwexam_inclasssign_number_int
6167 }
6168 }

```

(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.

```

6169 \newenvironment{assignment}[1][ ]{
6170 \__hwexam_assignment_args:n { #1 }
6171 %\sref@target
6172 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6173 \global\stepcounter{assignment}
6174 }{
6175 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6176 }
6177 \setcounter{problem}{0}
6178 \def\current@section@level{\document@hwexamtype}
6179 %\sref@label@id{\document@hwexamtype \thesection}
6180 \begin{@assignment}
6181 }{
6182 \end{@assignment}
6183 }

```

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

```

6184 \def\ass@title{
6185 \protect\document@hwexamtype~\arabic{assignment}
6186 \assignment@title{}\{;\}{} -- \given@due{}\}{}
6187 }
6188 \ifmultiple
6189 \newenvironment{@assignment}{
6190 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6191 \begin{omgroup}[loadmodules]{\ass@title}
6192 }{
6193 \begin{omgroup}{\ass@title}
6194 }
6195 }{
6196 \end{omgroup}
6197 }

```

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

```

6198 \else
6199 \newenvironment{@assignment}{
6200 \begin{center}\bf
6201 \Large@title\strut\\
6202 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
6203 \large\given@due{--;\}{}{;\}{}
6204 \end{center}
6205 }{}
6206 \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.

```

6207 \keys_define:nn { hwexam / inclassignment } {
6208 %id .str_set_x:N = \l__hwexam_assign_id_str,
6209 number .int_set:N = \l__hwexam_inclassign_number_int,
6210 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6211 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6212 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6213 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6214 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6215 }
6216 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6217 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6218 \tl_clear:N \l__hwexam_inclassign_title_tl
6219 \tl_clear:N \l__hwexam_inclassign_type_tl
6220 \tl_clear:N \l__hwexam_inclassign_given_tl
6221 \tl_clear:N \l__hwexam_inclassign_due_tl
6222 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6223 \keys_set:nn { hwexam / inclassignment }{ #1 }
6224 }
6225 \__hwexam_inclassignment_args:n {}
6226
6227 \newcommand\inputassignment[2][ ]{

```

```

6228 \_hwexam_inclassnment_args:n { #1 }
6229 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6230 \input{#2}
6231 }{
6232 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6233 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6234 }
6235 }
6236 \_hwexam_inclassnment_args:n {}
6237 }
6238 \newcommand\includeassignment[2][]{
6239 \newpage
6240 \inputassignment[#1]{#2}
6241 }

```

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

42.4 Typesetting Exams

\quizheading

```

6242 \ExplSyntaxOff
6243 \newcommand\quizheading[1]{%
6244 \def\@tas{#1}%
6245 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6246 \ifx\@tas\@empty\else%
6247 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6248 \fi%
6249 }
6250 \ExplSyntaxOn

```

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

\testheading

```

6251
6252 \def\hwexamheader{\input{hwexam-default.header}}
6253
6254 \def\hwexamminutes{
6255 \tl_if_empty:NTF \testheading@duration {
6256 {\testheading@min}~\hwexam@minutes@kw
6257 }{
6258 \testheading@duration
6259 }
6260 }
6261
6262 \keys_define:nn { hwexam / testheading } {
6263 min .tl_set:N = \testheading@min,
6264 duration .tl_set:N = \testheading@duration,
6265 reqpts .tl_set:N = \testheading@reqpts,
6266 tools .tl_set:N = \testheading@tools
6267 }
6268 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6269 \tl_clear:N \testheading@min
6270 \tl_clear:N \testheading@duration

```

```

6271 \tl_clear:N \testheading@reqpts
6272 \tl_clear:N \testheading@tools
6273 \keys_set:nn { hwexam / testheading }{ #1 }
6274 }
6275 \newenvironment{testheading}[1][]{
6276   \_hwexam_testheading_args:n{ #1 }
6277   \newcount\check@time\check@time=\testheading@min
6278   \advance\check@time by -\theassignment@totalmin
6279   \newif\if@bonuspoints
6280   \tl_if_empty:NTF \testheading@reqpts {
6281     \@bonuspointsfalse
6282   }{
6283     \newcount\bonus@pts
6284     \bonus@pts=\theassignment@totalpts
6285     \advance\bonus@pts by -\testheading@reqpts
6286     \edef\bonus@pts{\the\bonus@pts}
6287     \@bonuspointstrue
6288   }
6289   \edef\check@time{\the\check@time}
6290
6291   \makeatletter\hwexamheader\makeatother
6292 }{
6293   \newpage
6294 }

```

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

\testspace

```

6295 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6296 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6297 \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.

```

6298 <@=problems>
6299 \renewcommand\@problem[3]{
6300   \stepcounter{assignment@probs}
6301   \def\__problemspts{#2}
6302   \ifx\__problemspts\@empty\else
6303     \addtocounter{assignment@totalpts}{#2}
6304   \fi
6305   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6306   \xdef\correction@probs{\correction@probs & #1}%
6307   \xdef\correction@pts{\correction@pts & #2}
6308   \xdef\correction@reached{\correction@reached &}

```



```

6309 }
6310 <@@=hwexam>

```

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

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

```

6311 \newcounter{assignment@probs}
6312 \newcounter{assignment@totalpts}
6313 \newcounter{assignment@totalmin}
6314 \def\correction@probs{\correction@probs@kw}
6315 \def\correction@pts{\correction@pts@kw}
6316 \def\correction@reached{\correction@reached@kw}
6317 \stepcounter{assignment@probs}
6318 \newcommand\correction@table{
6319 \resizebox{\textwidth}{!}{%
6320 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6321 &\multicolumn{\theassignment@probs}{c|}||%|
6322 {\footnotesize\correction@forgrading@kw} &\\ \hline
6323 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6324 \correction@pts & \theassignment@totalpts & \\ \hline
6325 \correction@reached & & \[.7cm]\hline
6326 \end{tabular}}
6327 \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\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

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

```