

The sTeX3 Package *

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

2022-02-15

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 2022-02-15)

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	112
30	STeX-Symbols Implementation	117
30.1	Symbol Declarations	117
30.2	Notations	124
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	154
33	STeX-Statements Implementation	160
33.1	Definitions	160
33.2	Assertions	165
33.3	Examples	168
33.4	Logical Paragraphs	170

34 The Implementation	175
34.1 Package Options	175
34.2 Proofs	175
34.3 Justifications	181
35 \LaTeX-Others Implementation	183
36 \LaTeX-Metatheory Implementation	184
37 Tikzinput Implementation	187
38 document-structure.sty Implementation	189
38.1 The document-structure Class	189
38.2 Class Options	189
38.3 Beefing up the <code>document</code> environment	190
38.4 Implementation: document-structure Package	190
38.5 Package Options	190
38.6 Document Structure	192
38.7 Front and Backmatter	195
38.8 Global Variables	197
39 NotesSlides – Implementation	198
39.1 Class and Package Options	198
39.2 Notes and Slides	200
39.3 Header and Footer Lines	204
39.4 Frame Images	205
39.5 Colors and Highlighting	206
39.6 Sectioning	207
39.7 Excursions	209
40 The Implementation	211
40.1 Package Options	211
40.2 Problems and Solutions	212
40.3 Multiple Choice Blocks	218
40.4 Including Problems	219
40.5 Reporting Metadata	220
41 Implementation: The hwexam Class	222
41.1 Class Options	222
42 Implementation: The hwexam Package	224
42.1 Package Options	224
42.2 Assignments	225
42.3 Including Assignments	228
42.4 Typesetting Exams	229
42.5 Leftovers	231

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

```
\begin{smodule}{assoctest}
\symdef[ args=1 a ]{foo}{\comp{a:}#1\comp{; b:}#2\comp{; c:}#3}{\comp[#1\comp{;}# #1\comp{##2\comp{;#2\comp{}}}
$ \foo {w_1}{w_2}{x,y,z}$
\end{smodule}
```

Module 1: $a : w_1; b : w_2; c : [w_1; x + [w_1; y + z; w_2]; w_2]$

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

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

Module 2:
Module 3:
Module 4:

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 3

```
\begin{smodule}{ring}
\begin{copymodule}{group}{addition}
\renamedcl[name=universe]{universe}{runiverse}
\renamedcl[name=plus]{operation}{rplus}
\renamedcl[name=zero]{unit}{rzero}
\renamedcl[name=uminus]{inverse}{ruminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{ruminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\comp1}

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

Module 5:
Test: $a \cdot (c + d \cdot e)$

TODO: explain donotclone

Example 4

```
\begin{smodule}{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{smodule}
```

Module 6:

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` Both print this \TeX logo.
`\stex`

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.

Module 7: 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 5

```
\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 6

```
\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 7

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

```
\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 9

```
\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 10

```
\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 11

```
\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}
```

Module 8: 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 12

```
\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 13

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

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

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

5 6

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

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

Precedences

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

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

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

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

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

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

For example:

Module 9:

Example 14

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

```
 $a + b \cdot c$  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 ($\langle log\text{-}prefix \rangle *$) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

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

$\backslash\text{sTeX}$	Both print this sTeX logo.
$\backslash\text{stex}$	

$\backslash\text{stex_debug:nn}$	$\backslash\text{stex_debug:nn} \{ \langle log\text{-}prefix \rangle \} \{ \langle message \rangle \}$
-----------------------------------	---

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

$\backslash\text{stex_add_to_sms:n}$	Adds the provided code to the <code>.sms</code> -file of the document.
---	--

$\backslash\text{if@latexml}$	L ^A T _E X ₂ e and L ^A T _E X ₃ conditionals for L ^A T _E X _M L.
$\backslash\text{latexml_if_p:}$	
$\backslash\text{latexml_if:T}$	
$\backslash\text{latexml_if:F}$	
$\backslash\text{latexml_if:TF}$	

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

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

Annotates the HTML generated by $\langle content \rangle$ with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

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

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

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

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

Makes the macro $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the $\langle begin \rangle$ -code of the associated environments.

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

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

Chapter 10

sTEX-MathHub

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

10.1 Macros and Environments

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

10.1.1 Files, Paths, URIs

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

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

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

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

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:NTF</code>	\star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

10.1.2 MathHub Archives

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

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

Namespace 1:
file://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{smodule}{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{smodule}
\ExplSyntaxOff

```

```

Module 10:  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{smodule}[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{smodule}
\ExplSyntaxOff
```

Module 11: 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{smodule}{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{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{smodule}
```

Module 12:
Module 13:
Module 14: file://stextest?STEXModuleTest1
file://stextest?STEXModuleTest2
file://stextest?STEXModuleTest3
foo1
foo2
foo3

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

```
Module 15:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
              Meaning: >macro:->\protect \bar <
Module 16:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 17:    Meaning: >macro:->\stex_invoke_symbol:n {file://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{smodule}{UseTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:- \present\foo\
\end{smodule}
\begin{smodule}{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{smodule}

```

Module 18:
Module 19: Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}<
Module 20: Meaning: >undefined<
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory,file://stextest?UseTest3,file://stextest?UseTest2>
All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collection>, <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?mathematical-structure>, <http://mathhub.info/sTeX?Metatheory?dummyvar>, <http://mathhub.info/sTeX?Metatheory?collection>, <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?mathematical-structure>, <file://stextest?UseTest2?bar>

Test 10

```

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

```

Circular dependencies:
Module 21: >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[⟨args⟩]{⟨macroname⟩}</code>
-----------------------	--

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

```
\begin{smodule}{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{smodule}
```

```
Module 22:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://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{smodule}{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 23:

`\symdef`

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

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

Test 13

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

Module 24: $a + b + c$

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> .

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using `\withbrackets`.

`\withbrackets`

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX 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 `\left` and `\right` in display-mode.

Test 14

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

Module 25: $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 15

```
\begin{smodule}{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 $\mult{a,\plus{\frac{ab}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{c}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[{$\displaystyle
\mult{a,\plus{\frac{ab}{c}}}$}
\end{smodule}
```

Module 26: $\langle a \mid [b;c;d;e;f]^g \rangle$ and $\langle a \mid [b;c]^g \rangle$ and $\langle a \mid [b]^c \rangle$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

`\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{smodule}{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{smodule}
```

Module 27:
some a and some b and also some c here.
some *a* and some *b* and also some *c* here.
bar
or just some c
bar
or first b, then c, and finally a

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 16

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

18.2.6 Configuration of the Presentation

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

Environment	configuration macro	value
sproof	\spf@proof@kw	Proof
sketchproof	\spf@sketchproof@kw	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 `\pstmake@label@style` that takes two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the `\LaTeX \@for...:=...\do{...}` macro; see Figure 2 for examples.

style	example	configuration macro
long	0.8.1.5	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code>
angles	$\ggg 5$	<code>\def\pst@make@label@angles#1#2{\ensurermath{\@for\@I:=#1\do{\@I\angle}}#2}</code>
short	5	<code>\def\pst@make@label@short#1#2{#2}</code>
empty		<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 `sTeX` 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 \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX into a document format for mathematical knowledge management (MKM).

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

21.1 Introduction

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

The `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{smodule}{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 \TeX 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, \TeX 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[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets L^AT_EXML 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 L^AT_EX. 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 \TeX preamble of the course notes file. `\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` 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 \TeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `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 L^AT_EX source.

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

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

```

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

25.3 Messages and logging

```

42 <@@=stex_log>

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

\stex_debug:nn A simple macro issuing package messages with subpath.
53 \cs_new_protected:Nn \stex_debug:nn {
54   \clist_if_in:NnTF \c_stex_debug_clist { all } {
55     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
56       \\Debug~#1:~#2\\
57     }
58     \msg_none:nn{stex}{debug / #1}
59   }{
60     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
61       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
62         \\Debug~#1:~#2\\
63       }
64       \msg_none:nn{stex}{debug / #1}
65     }
66   }
67 }

```

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

Redirecting messages:

```

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

```

25.4 Persistence

77 `<@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

25.5 HTML Annotations

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

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

```

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

```

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

```

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

```



```

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

```

(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`

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

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

```

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

```

(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

```

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

```

(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.

```

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

```

```

186   }{
187     \par\rustex_annotate_HTML_end:
188   }
189 }{
190   \latexml_if:TF {
191     \cs_new_protected:Nn \stex_annotate:nnn {
192       \__stex_annotate_checkempty:n { #3 }
193       \mode_if_math:TF {
194         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
195           \tl_use:N \l__stex_annotate_arg_tl
196         }
197       }{
198         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
199           \tl_use:N \l__stex_annotate_arg_tl
200         }
201       }
202     }
203     \cs_new_protected:Nn \stex_annotate_invisible:n {
204       \__stex_annotate_checkempty:n { #1 }
205       \mode_if_math:TF {
206         \cs:w latexml@invisible@math\cs_end:{
207           \tl_use:N \l__stex_annotate_arg_tl
208         }
209       } {
210         \cs:w latexml@invisible@text\cs_end:{
211           \tl_use:N \l__stex_annotate_arg_tl
212         }
213       }
214     }
215     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
216       \__stex_annotate_checkempty:n { #3 }
217       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
218         \tl_use:N \l__stex_annotate_arg_tl
219       }
220     }
221     \NewDocumentEnvironment{stex_annotate_env} { m m } {
222       \par\begin{latexml@annotateenv}{#1}{#2}
223     }{
224       \par\end{latexml@annotateenv}
225     }
226   }{
227     \cs_new_protected:Nn \stex_annotate:nnn {#3}
228     \cs_new_protected:Nn \stex_annotate_invisible:n {}
229     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
230     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
231   }
232 }

```

(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

```

233 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\stex_do_aftergroup:nn`

```

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

```

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

```

300 </package>

```

Chapter 26

STEX -MathHub Implementation

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

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

```

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

```

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

```

(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`

```

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

```

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

```

```

368         }{
369         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
370         }
371     }
372     }{
373     \str_if_empty:NF \l_tmpa_tl {
374     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
375     }
376     }
377 }
378 }
379 \seq_gset_eq:NN #1 \l_tmpa_seq
380 }
381 }

```

(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}`

```

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

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(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`

```

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

```



```

406 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
407 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
408 \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

```

409 <@@=stex_files>

```

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

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

`\g_stex_files_stack` keeps track of file changes

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

414 \seq_gclear_new:N\g_stex_currentfile_seq
415 \cs_new_protected:Nn \stex_filestack_push:n {
416   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
417   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
418     \stex_path_from_string:Nn\g_stex_currentfile_seq{
419       \c_stex_pwd_str/#1
420     }
421   }
422   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
423   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
424 }
425 \cs_new_protected:Nn \stex_filestack_pop: {
426   \seq_if_empty:NF\g_stex_files_stack{
427     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
428   }
429   \seq_if_empty:NTF\g_stex_files_stack{
430     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
431   }{
432     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
433     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
434   }
435 }
436

```

```

437 \AddToHook{file/before}{
438   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
439 }
440 \AddToHook{file/after}{
441   \stex_filestack_pop:
442 }

```

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

26.4 MathHub Repositories

```

443 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
444 \str_if_empty:NTF\mathhub{
445   \stex_kpsewhich:n{-var-value~MATHHUB}
446   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
447
448   \str_if_empty:NTF\c_stex_mathhub_str{
449     \msg_warning:nn{stex}{warning/nomathhub}
450   }{
451     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
452     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
453   }
454 }{
455   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
456   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
457     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
458       \c_stex_pwd_str/\mathhub
459     }
460   }
461   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
462   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463 }

```

(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
464 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
465   \str_set:Nx \l_tmpa_str { #1 }
466   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
467     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
468     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
469     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
470     \__stex_mathhub_find_manifest:N \l_tmpa_seq
471     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
472       \msg_error:nnxx{stex}{error/norepository}{#1}{
473         \stex_path_to_string:N \c_stex_mathhub_str
474       }
475     } {
476       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
477     }
478   }
479 }

```

(End definition for _stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

480 \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:

```

481 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
482   \seq_set_eq:NN\l_tmpa_seq #1
483   \bool_set_true:N\l_tmpa_bool
484   \bool_while_do:Nn \l_tmpa_bool {
485     \seq_if_empty:NTF \l_tmpa_seq {
486       \bool_set_false:N\l_tmpa_bool
487     }{
488       \file_if_exist:nTF{
489         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
490       }{
491         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
492         \bool_set_false:N\l_tmpa_bool
493       }{
494         \file_if_exist:nTF{
495           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
496         }{
497           \seq_put_right:Nn\l_tmpa_seq{META-INF}
498           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
499           \bool_set_false:N\l_tmpa_bool
500         }{
501           \file_if_exist:nTF{
502             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
503           }{
504             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
505             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
506             \bool_set_false:N\l_tmpa_bool
507           }{
508             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
509           }
510         }
511       }
512     }
513   }
514   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
515 }

```

(End definition for _stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior File variable used for MANIFEST-files

516 \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:

```

517 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
518   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
519   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
520   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
521     \str_set:Nn \l_tmpa_str {##1}
522     \exp_args:NNoo \seq_set_split:Nnn
523       \l_tmpb_seq \c_colon_str \l_tmpa_str
524     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
525       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
526         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
527       }
528       \exp_args:No \str_case:nnTF \l_tmpa_tl {
529         {id} {
530           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531             { id } \l_tmpb_tl
532         }
533         {narration-base} {
534           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535             { narr } \l_tmpb_tl
536         }
537         {url-base} {
538           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539             { docurl } \l_tmpb_tl
540         }
541         {source-base} {
542           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543             { ns } \l_tmpb_tl
544         }
545         {ns} {
546           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547             { ns } \l_tmpb_tl
548         }
549         {dependencies} {
550           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
551             { deps } \l_tmpb_tl
552         }
553       }{}{}
554     }{}
555   }
556   \ior_close:N \c__stex_mathhub_manifest_ior
557 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

558 \cs_new_protected:Nn \stex_set_current_repository:n {
559   \stex_require_repository:n { #1 }
560   \prop_set_eq:Nc \l_stex_current_repository_prop {
561     c_stex_mathhub_#1_manifest_prop
562   }
563 }

```

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

`\stex_require_repository:n`

```

564 \cs_new_protected:Nn \stex_require_repository:n {
565   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
566     \stex_debug:nn{mathhub}{Opening~archive:~#1}
567     \__stex_mathhub_do_manifest:n { #1 }
568     \exp_args:Nx \stex_add_to_sms:n {
569       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
570         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
571         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
572         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
573         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
574       }
575     }
576   }
577 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

578 %\prop_new:N \l_stex_current_repository_prop
579
580 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
581 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
582   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
583 } {
584   \__stex_mathhub_parse_manifest:n { main }
585   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
586   \l_tmpa_str
587   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
588   \c_stex_mathhub_main_manifest_prop
589   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
590   \stex_debug:nn{mathhub}{Current~repository:~
591     \prop_item:Nn \l_stex_current_repository_prop {id}
592   }
593 }

```

(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.

```

594 \cs_new_protected:Nn \stex_in_repository:nn {
595   \str_set:Nx \l_tmpa_str { #1 }
596   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
597   \str_if_empty:NTF \l_tmpa_str {
598     \prop_if_exist:NTF \l_stex_current_repository_prop {
599       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
600       \exp_args:Ne \l_tmpa_cs{
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \l_tmpa_cs{}
605     }
606   }{
607     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

608 \stex_require_repository:n \l_tmpa_str
609 \str_set:Nx \l_tmpa_str { #1 }
610 \exp_args:Nne \use:nn {
611   \stex_set_current_repository:n \l_tmpa_str
612   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
613 }{
614   \stex_debug:nn{mathhub}{switching~back~to:~
615     \prop_if_exist:NTF \l_stex_current_repository_prop {
616       \prop_item:Nn \l_stex_current_repository_prop { id }::~
617       \meaning\l_stex_current_repository_prop
618     }{
619       no~repository
620     }
621   }
622   \prop_if_exist:NTF \l_stex_current_repository_prop {
623     \stex_set_current_repository:n {
624       \prop_item:Nn \l_stex_current_repository_prop { id }
625     }
626   }{
627     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
628   }
629 }
630 }
631 }

```

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

```

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

```

```

658     \stex_annotate_invisible:nnn{inputref}{
659       \l_tmpa_str / #2
660     }{}
661   }{
662     \begingroup
663       \inputreftrue
664       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
665     \endgroup
666   }
667 }
668 }
669
670 \NewDocumentCommand \inputref { 0{} m}{
671   \stex_inputref:nn{ #1 }{ #2 }
672 }
673
674 \cs_new_protected:Nn \stex_mhbibresource:nn {
675   \stex_in_repository:nn {#1} {
676     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
677   }
678 }
679 \newcommand\addmhbibresource[2][]{
680   \stex_mhbibresource:nn{ #1 }{ #2 }
681 }

```

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

`\mhpath`

```

682 \def \mhpath #1 #2 {
683   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
684     \c_stex_mathhub_str /
685     \prop_item:Nn \l_stex_current_repository_prop { id }
686     / source / #2
687   }{
688     \c_stex_mathhub_str / #1 / source / #2
689   }
690 }

```

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

`\libinput`

```

691 \cs_new_protected:Npn \libinput #1 {
692   \prop_if_exist:NF \l_stex_current_repository_prop {
693     \msg_error:nnn{stex}{error/notinarchive}\libinput
694   }
695   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
696     \msg_error:nnn{stex}{error/notinarchive}\libinput
697   }
698   \bool_set_false:N \l_tmpa_bool
699   \tl_clear:N \l_tmpa_tl
700   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
701   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
702   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
703   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

704 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
705 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
706 / meta-inf / lib / #1.tex}{
707 \bool_set_true:N \l_tmpa_bool
708 \tl_put_right:Nx \l_tmpa_tl {
709 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
710 / meta-inf / lib / #1.tex}
711 }
712 }{}
713 }
714 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
715 / \l_tmpa_str / lib / #1.tex
716 }{
717 \bool_set_true:N \l_tmpa_bool
718 \tl_put_right:Nx \l_tmpa_tl {
719 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
720 / \l_tmpa_str / lib / #1.tex}
721 }
722 }{}
723 \bool_if:NF \l_tmpa_bool {
724 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
725 }
726 \l_tmpa_tl
727 }

```

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

`\libusepackage`

```

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

```



```

754 \bool_if:NT \l_libusepackage_bool {
755   \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
756 }
757 \bool_set_true:N \l_libusepackage_bool
758 \tl_put_right:Nx \l_tmpa_tl {
759   \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
760     / \l_tmpa_str / lib / #2}
761 }
762 }{}
763 \bool_if:NF \l_libusepackage_bool {
764   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
765 }
766 \l_tmpa_tl
767 }

```

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

```

768
769 \AddToHook{begindocument}{
770 \ltx@ifpackageloaded{graphicx}{
771   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
772   \newcommand\mhgraphics[2][]{\%
773     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
774     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
775   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
776 }{}
777 \ltx@ifpackageloaded{listings}{
778   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
779   \newcommand\lstinputmhlsting[2][]{\%
780     \def\lst@mhrepos{}\setkeys{lst}{#1}%
781     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
782   \newcommand\clstinputmhlsting[2][]{\begin{center}\lstinputmhlsting[#1]{#2}\end{center}}
783 }{}
784 }
785
786
787 \end{package}

```

Chapter 27

STEX -References Implementation

```
788 <*package>
789
790 %%%%%%%%%%%%%% references.dtx %%%%%%%%%%%%%%
791
792 %\RequirePackage{hyperref}
793 %\RequirePackage{cleveref}
794 <@@=stex_refs>
795
796 Warnings and error messages
797
798 \iow_new:N \c__stex_refs_refs_iow
799 \AddToHook{begindocument}{
800   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
801 }
802 \AddToHook{enddocument}{
803   \iow_close:N \c__stex_refs_refs_iow
804 }
805
806 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
807
808 \NewDocumentCommand \STEXreftitle { m } {
809   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
810 }
811
```

27.1 Document URIs and URLs

```
809 \seq_new:N \g__stex_refs_all_refs_seq
810
811 \str_new:N \l_stex_current_docns_str
812
813 \cs_new_protected:Nn \stex_get_document_uri: {
814   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
815   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
816   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
817   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
818 }
819
```

```

818 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
819
820 \str_clear:N \l_tmpa_str
821 \prop_if_exist:NT \l_stex_current_repository_prop {
822   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824   }
825 }
826
827 \str_if_empty:NTF \l_tmpa_str {
828   \str_set:Nx \l_stex_current_docns_str {
829     file:/\stex_path_to_string:N \l_tmpa_seq
830   }
831 }{
832   \bool_set_true:N \l_tmpa_bool
833   \bool_while_do:Nn \l_tmpa_bool {
834     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
835     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
836       {source} { \bool_set_false:N \l_tmpa_bool }
837     }{}{
838       \seq_if_empty:NT \l_tmpa_seq {
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843
844   \seq_if_empty:NTF \l_tmpa_seq {
845     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
846   }{
847     \str_set:Nx \l_stex_current_docns_str {
848       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
849     }
850   }
851 }
852 }
853
854 \str_new:N \l_stex_current_docurl_str
855 \cs_new_protected:Nn \stex_get_document_url: {
856   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \str_clear:N \l_tmpa_str
863   \prop_if_exist:NT \l_stex_current_repository_prop {
864     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
865       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
866         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
867       }
868     }
869   }
870
871   \str_if_empty:NTF \l_tmpa_str {
872     \str_set:Nx \l_stex_current_docurl_str {

```

```

872     file:/\stex_path_to_string:N \l_tmpa_seq
873   }
874 }{
875   \bool_set_true:N \l_tmpa_bool
876   \bool_while_do:Nn \l_tmpa_bool {
877     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
878     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
879       {source} { \bool_set_false:N \l_tmpa_bool }
880     }{}{
881       \seq_if_empty:NT \l_tmpa_seq {
882         \bool_set_false:N \l_tmpa_bool
883       }
884     }
885   }
886
887   \seq_if_empty:NTF \l_tmpa_seq {
888     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
889   }{
890     \str_set:Nx \l_stex_current_docurl_str {
891       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
892     }
893   }
894 }
895 }

```

27.2 Setting Reference Targets

```

896 \str_const:Nn \c__stex_refs_url_str{URL}
897 \str_const:Nn \c__stex_refs_ref_str{REF}
898 % @currentlabel -> number
899 % @currentlabelname -> title
900 % @currentHref -> name.number <- id of some kind
901 % \theH# -> \arabic{section}
902 % \the# -> number
903 % \hyper@makecurrent{#}
904 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
905   \stex_get_document_uri:
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NT \l_tmpa_str {
908     \int_zero:N \l_tmpa_int
909     \bool_set_true:N \l_tmpa_bool
910     \bool_while_do:Nn \l_tmpa_bool {
911       \cs_if_exist:cTF {
912         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
913       }{
914         \int_incr:N \l_tmpa_int
915       }{
916         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
917         \bool_set_false:N \l_tmpa_bool
918       }
919     }
920   }
921   \str_set:Nx \l_tmpa_str {
922     \l_stex_current_docns_str??\l_tmpa_str

```

```

923 }
924 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
925 \stex_if_smsmode:TF {
926   \stex_get_document_url:
927   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
928   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
929 }{
930   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
931   \exp_args:Nx\label{sref_\l_tmpa_str}
932   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
933   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
934 }
935 }
936 \cs_new_protected:Npn \stexauxadddocref #1 {
937   \str_set:Nx \l_tmpa_str {#1}
938   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
939   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
940 }
941 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
942   \stex_get_document_uri:
943   \stex_if_smsmode:TF {
944     \stex_get_document_url:
945     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
946     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
947   }
948   }{
949     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
950     \exp_args:Nx\label{sref_sym_#1}
951   }
952   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
953   \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
954 }
955 }

```

27.3 Using References

```

956 \str_new:N \l__stex_refs_indocument_str
957 \keys_define:nn { stex / sref } {
958   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
959   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
960   pre           .tl_set:N = \l__stex_refs_pre_tl ,
961   post          .tl_set:N = \l__stex_refs_post_tl ,
962   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
963 }
964
965 \bool_new:N \c__stex_refs_hyperref_bool
966 \bool_set_false:N \c__stex_refs_hyperref_bool
967 \AddToHook{begindocument}{
968   \@ifpackageloaded{hyperref}{
969     \bool_set_true:N \c__stex_refs_hyperref_bool
970   }{}
971 }
972
973

```

```

974 \cs_new_protected:Nn \__stex_refs_args:n {
975   \tl_clear:N \l__stex_refs_linktext_tl
976   \tl_clear:N \l__stex_refs_fallback_tl
977   \tl_clear:N \l__stex_refs_pre_tl
978   \tl_clear:N \l__stex_refs_post_tl
979   \str_clear:N \l__stex_refs_repo_str
980   \keys_set:nn { stex / sref } { #1 }
981 }
982
983 \NewDocumentCommand \sref { 0{} m}{
984   \__stex_refs_args:n { #1 }
985   \str_if_empty:NTF \l__stex_refs_indocument_str {
986     \str_set:Nn \l_tmpa_str { #2 }
987     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
988     \tl_set:Nn \l_tmpa_tl {
989       \l__stex_refs_fallback_tl
990     }
991     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
992       \str_set:Nn \l_tmpb_str { ##1 }
993       \str_if_eq:eeT { \l_tmpa_str } {
994         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
995       } {
996         \seq_map_break:n {
997           \tl_set:Nn \l_tmpa_tl {
998             % doc uri in \l_tmpb_str
999             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
1000             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1001               % reference
1002               \cs_if_exist:cTF{autoref}{
1003                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1004               }{
1005                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1006               }
1007             }{
1008               % URL
1009               \if_bool:N \c__stex_refs_hyperref_bool {
1010                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1011               }{
1012                 \l__stex_refs_fallback_tl
1013               }
1014             }
1015           }
1016         }
1017       }
1018     }
1019     \l_tmpa_tl
1020   }{
1021     % TODO
1022   }
1023 }
1024
1025 \NewDocumentCommand \srefsym { 0{} m}{
1026   \stex_get_symbol:n { #2 }
1027   \__stex_refs_args:n { #1 }

```

```

1028 \str_if_empty:NTF \l__stex_refs_indocument_str {
1029   \tl_set:Nn \l_tmpa_tl {
1030     \l__stex_refs_fallback_tl
1031   }
1032   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1033     \tl_set:Nn \l_tmpa_tl {
1034       % doc uri in \l_tmpb_str
1035       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1036       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1037         % reference
1038         \cs_if_exist:cTF{autoref}{
1039           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1040         }{
1041           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1042         }
1043       }{
1044         % URL
1045         \if_bool:N \c__stex_refs_hyperref_bool {
1046           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1047         }{
1048           \l__stex_refs_fallback_tl
1049         }
1050       }
1051     }
1052   }
1053   \l_tmpa_tl
1054 }{
1055   % TODO
1056 }
1057 }
1058
1059 \cs_new_protected:Npn \srefsymuri #1 #2 {
1060   \hyperref[sref_sym_#1]{#2}
1061 }
1062
1063 </package>

```

Chapter 28

STEX -Modules Implementation

```
1064 <*package>
1065
1066 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1067
1068 <@@=stex_modules>
1069
1070 Warnings and error messages
1071 \msg_new:nnn{stex}{error/unknownmodule}{
1072   No~module~#1~found
1073 }
1074 \msg_new:nnn{stex}{error/syntax}{
1075   Syntax~error:~#1
1076 }
1077 \msg_new:nnn{stex}{error/siglanguage}{
1078   Module~#1~declares~signature~#2,~but~does~not~
1079   declare~its~language
1080 }
1081 \msg_new:nnn{stex}{error/conflictingmodules}{
1082   Conflicting~imports~for~module~#1
1083 }
1084
1085 \l_stex_current_module_str The current module:
1086 \str_new:N \l_stex_current_module_str
1087
1088 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1089
1090 \l_stex_all_modules_seq Stores all available modules
1091 \seq_new:N \l_stex_all_modules_seq
1092
1093 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1094
1095 \stex_if_in_module_p:
1096 \stex_if_in_module:TF
1097 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1098   \str_if_empty:NTF \l_stex_current_module_str
1099   \prg_return_false: \prg_return_true:
1100 }
```


(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`

```
1089 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1090   \prop_if_exist:cTF { c_stex_module_#1_prop }
1091   \prg_return_true: \prg_return_false:
1092 }
```

(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:

```
1093 \cs_new_protected:Nn \stex_add_to_current_module:n {
1094   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1095 }
1096 \cs_new_protected:Npn \STEXexport {
1097   \begingroup
1098   \newlinechar=-1\relax
1099   \endlinechar=-1\relax
1100   %\catcode'\ = 9\relax
1101   \expandafter\endgroup\STEXexport:n
1102 }
1103 \cs_new_protected:Nn \STEXexport:n {
1104   \ignorespaces #1
1105   \stex_add_to_current_module:n { \ignorespaces #1 }
1106   \stex_smsmode_do:
1107 }
1108 \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`

```
1109 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1110   \str_set:Nx \l_tmpa_str { #1 }
1111   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1112 }
1113
1114 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1115 % \str_set:Nx \l_tmpa_str { #1 }
1116 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_fields} { \l_tmpa_str }
1117 %}
```

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

`\stex_collect_imports:n`

```
1118 \cs_new_protected:Nn \stex_collect_imports:n {
1119   \seq_clear:N \l_stex_collect_imports_seq
1120   \__stex_modules_collect_imports:n {#1}
1121 }
1122 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1123   \seq_map_inline:cn {c_stex_module_#1_imports} {
1124     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1125       \__stex_modules_collect_imports:n { ##1 }
1126     }
1127 }
```

```

1127 }
1128 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1129   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1130 }
1131 }

```

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

`\stex_add_import_to_current_module:n`

```

1132 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1133   \str_set:Nx \l_tmpa_str { #1 }
1134   \exp_args:Nno
1135   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1136     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1137   }
1138 }

```

(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).

```

1139 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \seq_set_eq:NN \l_tmpa_seq #2
1142   % split off file extension
1143   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1144   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1145   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1146   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1147
1148   \bool_set_true:N \l_tmpa_bool
1149   \bool_while_do:Nn \l_tmpa_bool {
1150     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1151     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1152       {source} { \bool_set_false:N \l_tmpa_bool }
1153     }{}{
1154       \seq_if_empty:NT \l_tmpa_seq {
1155         \bool_set_false:N \l_tmpa_bool
1156       }
1157     }
1158   }
1159
1160   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1161   \str_if_empty:NTF \l_stex_modules_subpath_str {
1162     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1163   }{
1164     \str_set:Nx \l_stex_modules_ns_str {
1165       \l_tmpa_str/\l_stex_modules_subpath_str
1166     }
1167   }
1168 }

```

(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
1169 \str_new:N \l_stex_modules_ns_str
1170 \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.

```

1171 \cs_new_protected:Nn \stex_modules_current_namespace: {
1172   \str_clear:N \l_stex_modules_subpath_str
1173   \prop_if_exist:NTF \l_stex_current_repository_prop {
1174     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1175     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1176   }{
1177     % split off file extension
1178     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1179     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1180     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1182     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1183     \str_set:Nx \l_stex_modules_ns_str {
1184       file:/\stex_path_to_string:N \l_tmpa_seq
1185     }
1186   }
1187 }

```

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

28.1 The module environment

module arguments:

```

1188 \keys_define:nn { stex / module } {
1189   title      .tl_set:N      = \smodulename ,
1190   type       .str_set_x:N   = \smodulename ,
1191   id         .str_set_x:N   = \smoduleid ,
1192   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1193   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1194   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1195   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1196   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1197   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1198   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1199 }
1200
1201 \cs_new_protected:Nn \__stex_modules_args:n {
1202   \str_clear:N \smodulename
1203   \str_clear:N \smodulename
1204   \str_clear:N \smoduleid
1205   \str_clear:N \l_stex_module_ns_str
1206   \str_clear:N \l_stex_module_lang_str
1207   \str_clear:N \l_stex_module_sig_str
1208   \str_clear:N \l_stex_module_creators_str

```

```

1209 \str_clear:N \l_stex_module_contributors_str
1210 \str_clear:N \l_stex_module_meta_str
1211 \str_clear:N \l_stex_module_srccite_str
1212 \keys_set:nn { stex / module } { #1 }
1213 }
1214
1215 % module parameters here? In the body?
1216

```

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

```

1217 \cs_new_protected:Nn \stex_module_setup:nn {
1218   \str_set:Nx \l_stex_module_name_str { #2 }
1219   \__stex_modules_args:n { #1 }

   First, we set up the name and namespace of the module.
   Are we in a nested module?

1220   \stex_if_in_module:TF {
1221     % Nested module
1222     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1223       { ns } \l_stex_module_ns_str
1224     \str_set:Nx \l_stex_module_name_str {
1225       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1226         { name } / \l_stex_module_name_str
1227     }
1228   }{
1229     % not nested:
1230     \str_if_empty:NT \l_stex_module_ns_str {
1231       \stex_modules_current_namespace:
1232       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1233       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1234         / { \l_stex_module_ns_str }
1235       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1236       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1237         \str_set:Nx \l_stex_module_ns_str {
1238           \stex_path_to_string:N \l_tmpa_seq
1239         }
1240       }
1241     }
1242   }

```

Next, we determine the language of the module:

```

1243 \str_if_empty:NT \l_stex_module_lang_str {
1244   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1245   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1246   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1247   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1248   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1249     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1250       inferred~from~file~name}
1251     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1252   }
1253 }
1254
1255 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1256 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1257 \l_tmpa_str {
1258   \ltx@ifpackageloaded{babel}{
1259     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1260   }{}
1261 } {
1262   \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1263 }
1264 }}

```

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

```

1265 \str_if_empty:NTF \l_stex_module_sig_str {
1266   \exp_args:Nnx \prop_gset_from_keyval:cn {
1267     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1268   } {
1269     name      = \l_stex_module_name_str ,
1270     ns        = \l_stex_module_ns_str ,
1271     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1272     lang      = \l_stex_module_lang_str ,
1273     sig       = \l_stex_module_sig_str ,
1274     meta      = \l_stex_module_meta_str
1275   }
1276   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1277   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1278   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1279   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1280   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1281 \str_if_empty:NT \l_stex_module_meta_str {
1282   \str_set:Nx \l_stex_module_meta_str {
1283     \c_stex_metatheory_ns_str ? Metatheory
1284   }
1285 }
1286 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1287   \bool_set_true:N \l_stex_in_meta_bool
1288   \exp_args:Nx \stex_add_to_current_module:n {
1289     \bool_set_true:N \l_stex_in_meta_bool
1290     \stex_activate_module:n {\l_stex_module_meta_str}
1291     \bool_set_false:N \l_stex_in_meta_bool
1292   }
1293   \stex_activate_module:n {\l_stex_module_meta_str}
1294   \bool_set_false:N \l_stex_in_meta_bool
1295 }
1296 }{
1297   \str_if_empty:NT \l_stex_module_lang_str {
1298     \msg_error:nxxx{stex}{error/siglanguage}{
1299       \l_stex_module_ns_str?\l_stex_module_name_str
1300     }{\l_stex_module_sig_str}
1301   }
1302
1303   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1304   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1305 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1306 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1307 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1308 \str_set:Nx \l_tmpa_str {
1309   \stex_path_to_string:N \l_tmpa_seq /
1310   \l_tmpa_str . \l_stex_module_sig_str .tex
1311 }
1312 \IfFileExists \l_tmpa_str {
1313   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1314     \str_clear:N \l_stex_current_module_str
1315     \seq_clear:N \l_stex_all_modules_seq
1316     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1317   }
1318 }{
1319   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1320 }
1321 \stex_if_smsmode:F {
1322   \stex_activate_module:n {
1323     \l_stex_module_ns_str ? \l_stex_module_name_str
1324   }
1325 }
1326 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1327 }
1328 }

```

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

module The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1329 \int_new:N \l_stex_module_group_depth_int
1330 \cs_new_protected:Nn \__stex_modules_begin_module: {
1331   \stex_reactivate_macro:N \STEXexport
1332   \stex_reactivate_macro:N \importmodule
1333   \stex_reactivate_macro:N \symdecl
1334   \stex_reactivate_macro:N \notation
1335   \stex_reactivate_macro:N \symdef
1336
1337   \stex_debug:nn{modules}{
1338     New~module:\\
1339     Namespace:~\l_stex_module_ns_str\\
1340     Name:~\l_stex_module_name_str\\
1341     Language:~\l_stex_module_lang_str\\
1342     Signature:~\l_stex_module_sig_str\\
1343     Metatheory:~\l_stex_module_meta_str\\
1344     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1345   }
1346
1347   \seq_put_right:Nx \l_stex_all_modules_seq {
1348     \l_stex_module_ns_str ? \l_stex_module_name_str
1349   }
1350
1351   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1352   %   { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1353
1354
1355 \stex_if_smsmode:F{
1356   \begin{stex_annotate_env} {theory} {
1357     \l_stex_module_ns_str ? \l_stex_module_name_str
1358   }
1359
1360   \stex_annotate_invisible:nnn{header}{} {
1361     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1362     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1363     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1364       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1365     }
1366     \str_if_empty:NF \smodulotype {
1367       \stex_annotate:nnn{type}{\smodulotype}{}
1368     }
1369   }
1370 }
1371 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1372 % TODO: Inherit metatheory for nested modules?
1373 }
1374 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for _stex_modules_begin_module:.)

```

\_stex_modules_end_module: implements \end{module}

1375 \cs_new_protected:Nn \_stex_modules_end_module: {
1376 % \str_set:Nx \l_tmpa_str {
1377 %   c_stex_module_
1378 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1379 %   \prop_item:Nn \l_stex_current_module_prop { name }
1380 %   _prop
1381 % }
1382 %^^A \prop_new:c { \l_tmpa_str }
1383 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1384 \stex_debug:nn{modules}{Closing-module-\prop_item:cn {c_stex_module_\l_stex_current_module_
1385 }

```

(End definition for _stex_modules_end_module:.)

smodule The core environment, with no header

```

1386 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1387 \NewDocumentEnvironment { smodule } { 0{} m } {
1388   \stex_module_setup:nn{#1}{#2}
1389   \par
1390   \stex_if_smsmode:F{
1391     \tl_clear:N \l_tmpa_tl
1392     \clist_map_inline:Nn \smodulotype {
1393       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1394         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1395       }
1396     }
1397     \tl_if_empty:NTF \l_tmpa_tl {
1398       \_stex_modules_smodule_start:

```

```

1399     }{
1400         \l_tmpa_tl
1401     }
1402 }
1403 \__stex_modules_begin_module:
1404 \stex_ref_new_doc_target:n \smoduleid
1405 \stex_smsmode_do:
1406 } {
1407 \__stex_modules_end_module:
1408 \stex_if_smsmode:TF {
1409 %     \exp_args:Nx \stex_add_to_sms:n {
1410 %         \prop_gset_from_keyval:cn {
1411 %             c_stex_module_
1412 %             \prop_item:Nn \l_stex_current_module_prop { ns } ?
1413 %             \prop_item:Nn \l_stex_current_module_prop { name }
1414 %             _prop
1415 %         } {
1416 %             name      = \prop_item:cn { \l_tmpa_str } { name } ,
1417 %             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1418 %             file      = \prop_item:cn { \l_tmpa_str } { file } ,
1419 %             lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1420 %             sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1421 %             meta      = \prop_item:cn { \l_tmpa_str } { meta }
1422 %         }
1423 %     }
1424 }{
1425     \end{stex_annotate_env}
1426     \clist_set:No \l_tmpa_clist \smoduletype
1427     \tl_clear:N \l_tmpa_tl
1428     \clist_map_inline:Nn \l_tmpa_clist {
1429         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1430             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1431         }
1432     }
1433     \tl_if_empty:NTF \l_tmpa_tl {
1434         \__stex_modules_smodule_end:
1435     }{
1436         \l_tmpa_tl
1437     }
1438 }
1439 }
1440
1441 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1442 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1443
1444 \newcommand\stexpatchmodule[3] [] {
1445     \str_set:Nx \l_tmpa_str{ #1 }
1446     \str_if_empty:NTF \l_tmpa_str {
1447         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1448         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1449     }{
1450         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1451         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1452     }

```



```

1453 }
1454

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1455 \NewDocumentCommand \STEXModule { m } {
1456   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1457   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1458   \tl_set:Nn \l_tmpa_tl {
1459     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1460   }
1461   \seq_map_inline:Nn \l_stex_all_modules_seq {
1462     \str_set:Nn \l_tmpb_str { ##1 }
1463     \str_if_eq:eeT { \l_tmpa_str } {
1464       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1465     } {
1466       \seq_map_break:n {
1467         \tl_set:Nn \l_tmpa_tl {
1468           \stex_invoke_module:n { ##1 }
1469         }
1470       }
1471     }
1472   }
1473   \l_tmpa_tl
1474 }
1475
1476 \cs_new_protected:Nn \stex_invoke_module:n {
1477   \stex_debug:nn{modules}{Invoking~module~#1}
1478   \peek_charcode_remove:NTF ! {
1479     \__stex_modules_invoke_uri:nN { #1 }
1480   } {
1481     \peek_charcode_remove:NTF ? {
1482       \__stex_modules_invoke_symbol:nn { #1 }
1483     } {
1484       \msg_error:nnx{stex}{error/syntax}{
1485         ?~or~!~expected~after~
1486         \c_backslash_str STEXModule{#1}
1487       }
1488     }
1489   }
1490 }
1491
1492 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1493   \str_set:Nn #2 { #1 }
1494 }
1495
1496 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1497   \stex_invoke_symbol:n{#1?#2}
1498 }

```

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

`\stex_activate_module:n`

```
1499 \bool_new:N \l_stex_in_meta_bool
1500 \bool_set_false:N \l_stex_in_meta_bool
1501 \cs_new_protected:Nn \stex_activate_module:n {
1502   \stex_debug:nn{modules}{Activating~module~#1}
1503   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1504     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1505   }
1506   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1507     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1508     \use:c{ c_stex_module_#1_code }
1509   }
1510 }
```

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

```
1511 \</package>
```

Chapter 29

STEX -Module Inheritance Implementation

```
1512 <*package>
1513
1514 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1515
```

29.1 SMS Mode

```
1516 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1517 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1518 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1519 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1520
1521 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1522   \makeatletter
1523   \makeatother
1524   \ExplSyntaxOn
1525   \ExplSyntaxOff
1526   \rustexBREAK
1527 }
1528
1529 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1530   \symdef
1531   \importmodule
1532   \notation
1533   \symdecl
1534   \STEXexport
1535   \inlineass
1536   \inlinedef
1537   \inlineex
1538   \endinput
1539 }
```

```

1540
1541 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1542   \tl_to_str:n {
1543     smodule,
1544     copymodule,
1545     interpretmodule
1546     sdefinition,
1547     sexample,
1548     sassertion,
1549     sparagraph
1550   }
1551 }

```

(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
1552 \bool_new:N \g__stex_smsmode_bool
1553 \bool_set_false:N \g__stex_smsmode_bool
1554 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1555   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1556 }

```

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

`\stex_in_smsmode:nn`

```

1557 \cs_new_protected:Nn \stex_in_smsmode:nn {
1558   \vbox_set:Nn \l_tmpa_box {
1559     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1560     \bool_gset_true:N \g__stex_smsmode_bool
1561     #2
1562     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1563   }
1564   \box_clear:N \l_tmpa_box
1565 }
1566
1567 \quark_new:N \q__stex_smsmode_break
1568
1569 %\ior_new:N \c__stex_smsmode_ior
1570 %\tl_new:N \l__stex_smsmode_filecontent_tl
1571 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1572   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1573   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1574   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1575     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1576   % }
1577   % \ior_close:N \c__stex_smsmode_ior
1578   \stex_filestack_push:n{#1}
1579   \stex_in_smsmode:nn{#1} {
1580     #2
1581     \everyeof{\q__stex_smsmode_break\noexpand}
1582     \expandafter\expandafter\expandafter
1583     \stex_smsmode_do:
1584     \csname @ @ input\endcsname "#1"\relax
1585     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl

```

```

1586 }
1587 \stex_filestack_pop:
1588 }

```

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

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

```

1589 \cs_new_protected:Npn \stex_smsmode_do: {
1590   \stex_if_smsmode:T {
1591     \__stex_smsmode_do:w
1592   }
1593 }
1594 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1595   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1596     \expandafter\if\expandafter\relax\noexpand#1
1597     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1598   } \else\expandafter\__stex_smsmode_do:w\fi
1599 }{
1600   \__stex_smsmode_do:w %#1
1601 }
1602 }
1603 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1604   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1605     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1606       #1\__stex_smsmode_do:w
1607     }{
1608       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1609         #1
1610       }{
1611         \cs_if_eq:NNTF \begin #1 {
1612           \__stex_smsmode_check_begin:n
1613         }{
1614           \cs_if_eq:NNTF \end #1 {
1615             \__stex_smsmode_check_end:n
1616           }{
1617             \__stex_smsmode_do:w
1618           }
1619         }
1620       }
1621     }
1622   }
1623 }
1624
1625 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1626   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1627     \begin{#1}
1628   }{
1629     \__stex_smsmode_do:w
1630   }
1631 }
1632 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1633   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1634     \end{#1}\__stex_smsmode_do:w

```

```

1635   }{
1636     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1637   }
1638 }

```

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

29.2 Inheritance

```

1639 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1640 \cs_new_protected:Nn \stex_import_module_uri:nn {
1641   \str_set:Nx \l_stex_import_archive_str { #1 }
1642   \str_set:Nn \l_stex_import_path_str { #2 }
1643
1644   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1645   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1646   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1647
1648   \stex_modules_current_namespace:
1649   \bool_lazy_all:nTF {
1650     {\str_if_empty_p:N \l_stex_import_archive_str}
1651     {\str_if_empty_p:N \l_stex_import_path_str}
1652     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1653   }{
1654     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1655     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1656   }{
1657     \str_if_empty:NT \l_stex_import_archive_str {
1658       \prop_if_exist:NT \l_stex_current_repository_prop {
1659         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1660       }
1661     }
1662     \str_if_empty:NTF \l_stex_import_archive_str {
1663       \str_if_empty:NF \l_stex_import_path_str {
1664         \str_set:Nx \l_stex_import_ns_str {
1665           \l_stex_module_ns_str / \l_stex_import_path_str
1666         }
1667       }
1668     }{
1669       \stex_require_repository:n \l_stex_import_archive_str
1670       \prop_get:cnN { c_stex_mathhub\l_stex_import_archive_str _manifest_prop } { ns }
1671       \l_stex_import_ns_str
1672       \str_if_empty:NF \l_stex_import_path_str {
1673         \str_set:Nx \l_stex_import_ns_str {
1674           \l_stex_import_ns_str / \l_stex_import_path_str
1675         }
1676       }
1677     }
1678   }
1679 }

```

(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 1680 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1681 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1682 \str_new:N \l_stex_import_path_str
1683 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1684 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1685   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1686
1687     % archive
1688     \str_set:Nx \l_tmpa_str { #2 }
1689     \str_if_empty:NTF \l_tmpa_str {
1690       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1691     } {
1692       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1693       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1694       \seq_put_right:Nn \l_tmpa_seq { source }
1695     }
1696
1697     % path
1698     \str_set:Nx \l_tmpb_str { #3 }
1699     \str_if_empty:NTF \l_tmpb_str {
1700       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1701
1702       \ltx@ifpackageloaded{babel} {
1703         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1704           { \language } \l_tmpb_str {
1705           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1706         }
1707       } {
1708         \str_clear:N \l_tmpb_str
1709       }
1710
1711       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1712       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1713         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1714       }{
1715         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1716         \IfFileExists{ \l_tmpa_str.tex }{
1717           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1718         }{
1719           % try english as default
1720           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1721           \IfFileExists{ \l_tmpa_str.en.tex }{
1722             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1723           }{
1724             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1725           }
1726         }
1727       }
1728

```

```

1729 } {
1730   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1731   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1732
1733   \ltx@ifpackageloaded{babel} {
1734     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1735       { \language } \l_tmpb_str {
1736       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1737     }
1738   } {
1739     \str_clear:N \l_tmpb_str
1740   }
1741
1742   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1743
1744   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1745   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1746     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1747   }{
1748     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1749     \IfFileExists{ \l_tmpa_str/#4.tex }{
1750       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1751     }{
1752       % try english as default
1753       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1754       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1755         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1756       }{
1757         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1758         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1759           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1760         }{
1761           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1762           \IfFileExists{ \l_tmpa_str.tex }{
1763             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1764           }{
1765             % try english as default
1766             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1767             \IfFileExists{ \l_tmpa_str.en.tex }{
1768               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1769             }{
1770               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1771             }
1772           }
1773         }
1774       }
1775     }
1776   }
1777 }
1778
1779 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1780   \seq_clear:N \l_stex_all_modules_seq
1781   \str_clear:N \l_stex_current_module_str
1782   \str_set:Nx \l_tmpb_str { #2 }

```



```

1783     \str_if_empty:NF \l_tmpb_str {
1784       \stex_set_current_repository:n { #2 }
1785     }
1786     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1787   }
1788
1789   \stex_if_module_exists:nF { #1 ? #4 } {
1790     \msg_error:nnx{stex}{error/unknownmodule}{
1791       #1?#4~(in~file~\g__stex_importmodule_file_str)
1792     }
1793   }
1794 }
1795 \stex_activate_module:n { #1 ? #4 }
1796 }

```

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

`\importmodule`

```

1797 \NewDocumentCommand \importmodule { 0{} m } {
1798   \stex_import_module_uri:nn { #1 } { #2 }
1799   \stex_debug:nn{modules}{Importing~module:~
1800     \l_stex_import_ns_str ? \l_stex_import_name_str
1801   }
1802   \stex_if_smsmode:F {
1803     \stex_import_require_module:nnnn
1804     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1805     { \l_stex_import_path_str } { \l_stex_import_name_str }
1806     \stex_annotate_invisible:nnn
1807     {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1808   }
1809   \exp_args:Nx \stex_add_to_current_module:n {
1810     \stex_import_require_module:nnnn
1811     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1812     { \l_stex_import_path_str } { \l_stex_import_name_str }
1813   }
1814   \exp_args:Nx \stex_add_import_to_current_module:n {
1815     \l_stex_import_ns_str ? \l_stex_import_name_str
1816   }
1817   \stex_smsmode_do:
1818 }
1819 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1820 \NewDocumentCommand \usemodule { 0{} m } {
1821   \stex_if_smsmode:F {
1822     \stex_import_module_uri:nn { #1 } { #2 }
1823     \stex_import_require_module:nnnn
1824     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1825     { \l_stex_import_path_str } { \l_stex_import_name_str }
1826     \stex_annotate_invisible:nnn
1827     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1828   }
1829   \stex_smsmode_do:

```

1830 }

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

1831 </package>

Chapter 30

STEX -Symbols Implementation

```
1832 <*package>
1833
1834 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1835
```

Warnings and error messages

```
1836
```

30.1 Symbol Declarations

```
1837 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1838 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1839 \NewDocumentCommand \STEXsymbol { m } {
1840   \stex_get_symbol:n { #1 }
1841   \exp_args:No
1842   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1843 }
```

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

symdecl arguments:

```
1844 \keys_define:nn { stex / symdecl } {
1845   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1846   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1847   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1848   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1849   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1850   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1851   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1852   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1853 }
```

```

1854
1855 \bool_new:N \l_stex_symdecl_make_macro_bool
1856
1857 \cs_new_protected:Nn \__stex_symdecl_args:n {
1858   \str_clear:N \l_stex_symdecl_name_str
1859   \str_clear:N \l_stex_symdecl_args_str
1860   \bool_set_false:N \l_stex_symdecl_local_bool
1861   \tl_clear:N \l_stex_symdecl_type_tl
1862   \tl_clear:N \l_stex_symdecl_definiens_tl
1863
1864   \keys_set:nn { stex / symdecl } { #1 }
1865 }

```

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

```

1866
1867 \NewDocumentCommand \symdecl { s O{} m } {
1868   \__stex_symdecl_args:n { #2 }
1869   \IfBooleanTF #1 {
1870     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1871   } {
1872     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1873   }
1874   \stex_symdecl_do:n { #3 }
1875   \stex_smsmode_do:
1876 }
1877 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1878 \cs_new_protected:Nn \stex_symdecl_do:n {
1879   \stex_if_in_module:F {
1880     % TODO throw error? some default namespace?
1881   }
1882
1883   \str_if_empty:NT \l_stex_symdecl_name_str {
1884     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1885   }
1886
1887   \prop_if_exist:cT { l_stex_symdecl_
1888     \l_stex_current_module_str ?
1889     \l_stex_symdecl_name_str
1890   }_prop
1891   }{
1892     % TODO throw error (beware of circular dependencies)
1893   }
1894
1895   \prop_clear:N \l_tmpa_prop
1896   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1897   \seq_clear:N \l_tmpa_seq
1898   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1899   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1900

```

```

1901 \exp_args:No \stex_add_constant_to_current_module:n {
1902   \l_stex_symdecl_name_str
1903 }
1904
1905 % arity/args
1906 \int_zero:N \l_tmpb_int
1907
1908 \bool_set_true:N \l_tmpa_bool
1909 \str_map_inline:Nn \l_stex_symdecl_args_str {
1910   \token_case_meaning:NnF ##1 {
1911     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1912     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1913     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1914     {\tl_to_str:n a} {
1915       \bool_set_false:N \l_tmpa_bool
1916       \int_incr:N \l_tmpb_int
1917     }
1918     {\tl_to_str:n B} {
1919       \bool_set_false:N \l_tmpa_bool
1920       \int_incr:N \l_tmpb_int
1921     }
1922   }{
1923     \msg_set:nnn{stex}{error/wrongargs}{
1924       args~value~in~symbol~declaration~for~
1925       \l_stex_current_module_str ?
1926       \l_stex_symdecl_name_str ~
1927       needs~to~be~
1928       i,~a,~b~or~B,~but~##1~given
1929     }
1930     \msg_error:nn{stex}{error/wrongargs}
1931   }
1932 }
1933 \bool_if:NTF \l_tmpa_bool {
1934   % possibly numeric
1935   \str_if_empty:NTF \l_stex_symdecl_args_str {
1936     \prop_put:Nnn \l_tmpa_prop { args } {}
1937     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1938   }{
1939     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1940     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1941     \str_clear:N \l_tmpa_str
1942     \int_step_inline:nn \l_tmpa_int {
1943       \str_put_right:Nn \l_tmpa_str i
1944     }
1945     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1946   }
1947 } {
1948   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1949   \prop_put:Nnx \l_tmpa_prop { arity }
1950     { \str_count:N \l_stex_symdecl_args_str }
1951 }
1952 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1953
1954

```

```

1955 % semantic macro
1956
1957 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1958   \exp_args:Nx \stex_do_aftergroup:n {
1959     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1960       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1961     }}
1962   }
1963
1964   \bool_if:NF \l_stex_symdecl_local_bool {
1965     \exp_args:Nx \stex_add_to_current_module:n {
1966       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1967         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1968       } }
1969     }
1970   }
1971 }
1972
1973 % add to all symbols
1974
1975 \bool_if:NF \l_stex_symdecl_local_bool {
1976   \exp_args:Nx \stex_add_to_current_module:n {
1977     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1978       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979     }
1980   }
1981 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1982 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1983 %   }
1984 }
1985
1986 \stex_debug:nn{symbols}{New~symbol:~
1987   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1988   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1989   Args:~\prop_item:Nn \l_tmpa_prop { args }
1990 }
1991
1992 % circular dependencies require this:
1993
1994 \prop_if_exist:cF {
1995   l_stex_symdecl_
1996   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1997   _prop
1998 } {
1999   \prop_set_eq:cN {
2000     l_stex_symdecl_
2001     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002     _prop
2003   } \l_tmpa_prop
2004 }
2005
2006 \seq_clear:c {
2007   l_stex_symdecl_
2008   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2009   _notations
2010 }
2011
2012 \bool_if:NF \l_stex_symdecl_local_bool {
2013   \exp_args:Nx
2014   \stex_add_to_current_module:n {
2015     \seq_clear:c {
2016       l_stex_symdecl_
2017       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018       _notations
2019     }
2020     \prop_set_from_keyval:cn {
2021       l_stex_symdecl_
2022       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2023       _prop
2024     } {
2025       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2026       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2027       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2028       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2029       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2030       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2031     }
2032   }
2033 }
2034
2035 \stex_if_smsmode:TF {
2036   \bool_if:NF \l_stex_symdecl_local_bool {
2037     % \exp_args:Nx \stex_add_to_sms:n {
2038     %   \prop_set_from_keyval:cn {
2039     %     l_stex_symdecl_
2040     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041     %     _prop
2042     %   } {
2043     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2044     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2045     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2046     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2047     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2048     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2049     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2050     %   }
2051     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2052     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2053     %   }
2054     % }
2055   }
2056 }{
2057   \exp_args:Nx \stex_do_aftergroup:n {
2058     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2059       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2060     }
2061   }
2062   \stex_if_do_html:T {

```

```

2063 \stex_annotate_invisible:nnn {symdecl} {
2064   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2065 } {
2066   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2067   \stex_annotate_invisible:nnn{args}{}{
2068     \prop_item:Nn \l_tmpa_prop { args }
2069   }
2070   \stex_annotate_invisible:nnn{macroname}{#1}{}
2071   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2072     \stex_annotate_invisible:nnn{definiens}{}
2073     { $\l_stex_symdecl_definiens_tl$ }
2074   }
2075 }
2076 }
2077 }
2078 }

```

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

`\stex_get_symbol:n`

```

2079 \str_new:N \l_stex_get_symbol_uri_str
2080
2081 \cs_new_protected:Nn \stex_get_symbol:n {
2082   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2083     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2084   }{
2085     % argument is a string
2086     % is it a command name?
2087     \cs_if_exist:cTF { #1 }{
2088       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2089       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2090       \str_if_empty:NNTF \l_tmpa_str {
2091         \exp_args:Nx \cs_if_eq:NNTF {
2092           \tl_head:N \l_tmpa_tl
2093         } \stex_invoke_symbol:n {
2094           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2095         }{
2096           \__stex_symdecl_get_symbol_from_string:n { #1 }
2097         }
2098       } {
2099         \__stex_symdecl_get_symbol_from_string:n { #1 }
2100       }
2101     }{
2102       % argument is not a command name
2103       \__stex_symdecl_get_symbol_from_string:n { #1 }
2104       % \l_stex_all_symbols_seq
2105     }
2106   }
2107 }
2108
2109 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2110   \str_set:Nn \l_tmpa_str { #1 }
2111   \bool_set_false:N \l_tmpa_bool
2112   \stex_if_in_module:T {

```



```

2113 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2114 \bool_set_true:N \l_tmpa_bool
2115 \str_set:Nx \l_stex_get_symbol_uri_str {
2116 \l_stex_current_module_str ? #1
2117 }
2118 }
2119 }
2120 \bool_if:NF \l_tmpa_bool {
2121 \tl_set:Nn \l_tmpa_tl {
2122 \msg_set:nnn{stex}{error/unknownsymbol}{
2123 No~symbol~#1~found!
2124 }
2125 \msg_error:nn{stex}{error/unknownsymbol}
2126 }
2127 \str_set:Nn \l_tmpa_str { #1 }
2128 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2129 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2130 \str_set:Nn \l_tmpb_str { ##1 }
2131 \str_if_eq:eeT { \l_tmpa_str } {
2132 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2133 } {
2134 \seq_map_break:n {
2135 \tl_set:Nn \l_tmpa_tl {
2136 \str_set:Nn \l_stex_get_symbol_uri_str {
2137 ##1
2138 }
2139 }
2140 }
2141 }
2142 }
2143 \l_tmpa_tl
2144 }
2145 }
2146
2147 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2148 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2149 { \tl_tail:N \l_tmpa_tl }
2150 \tl_if_single:NTF \l_tmpa_tl {
2151 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2152 \exp_after:wN \str_set:Nn \exp_after:wN
2153 \l_stex_get_symbol_uri_str \l_tmpa_tl
2154 }{
2155 % TODO
2156 % tail is not a single group
2157 }
2158 }{
2159 % TODO
2160 % tail is not a single group
2161 }
2162 }

```

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

30.2 Notations

```

2163 <@@=stex_notation>

notation arguments:
2164 \keys_define:nn { stex / notation } {
2165   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2166   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2167   prec .str_set_x:N = \l__stex_notation_prec_str ,
2168   op .tl_set:N = \l__stex_notation_op_tl ,
2169   primary .bool_set:N = \l__stex_notation_primary_bool ,
2170   primary .default:n = {true} ,
2171   unknown .code:n = \str_set:Nx
2172     \l__stex_notation_variant_str \l_keys_key_str
2173 }
2174
2175 \cs_new_protected:Nn \stex_notation_args:n {
2176   \str_clear:N \l__stex_notation_lang_str
2177   \str_clear:N \l__stex_notation_variant_str
2178   \str_clear:N \l__stex_notation_prec_str
2179   \tl_clear:N \l__stex_notation_op_tl
2180   \bool_set_false:N \l__stex_notation_primary_bool
2181
2182   \keys_set:nn { stex / notation } { #1 }
2183 }

```

\notation

```

2184 \NewDocumentCommand \notation { 0{ } m } {
2185   \stex_notation_args:n { #1 }
2186   \tl_clear:N \l_stex_symdecl_definiens_tl
2187   \stex_get_symbol:n { #2 }
2188   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2189 }
2190 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2191 \seq_new:N \l__stex_notation_precedences_seq
2192 \tl_new:N \l__stex_notation_opprec_tl
2193 \int_new:N \l__stex_notation_currarg_int
2194
2195 \cs_new_protected:Nn \stex_notation_do:nn {
2196   \let\l_stex_current_symbol_str\relax
2197   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2198   \seq_clear:N \l__stex_notation_precedences_seq
2199   \tl_clear:N \l__stex_notation_opprec_tl
2200   \prop_get:cnN {
2201     l_stex_symdecl_ #1 _prop
2202   } { args } \l__stex_notation_args_str
2203
2204   % precedences
2205   \prop_get:cnN {
2206     l_stex_symdecl_ #1 _prop
2207   } { arity } \l__stex_notation_arity_str

```

```

2208 \str_if_empty:NTF \l__stex_notation_prec_str {
2209   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2210     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2211   }{
2212     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2213   }
2214 } {
2215   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2216     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2217     \int_step_inline:nn { \l__stex_notation_arity_str } {
2218       \exp_args:NNo
2219       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infpref }
2220     }
2221   }{
2222     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2223     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2224       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2225       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2226         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2227         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2228         \seq_map_inline:Nn \l_tmpa_seq {
2229           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2230         }
2231       }
2232     }{
2233       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2234         \tl_set:No \l__stex_notation_opprec_tl { \infpref }
2235       }{
2236         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2237       }
2238     }
2239   }
2240 }
2241
2242 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2243 \int_step_inline:nn { \l__stex_notation_arity_str } {
2244   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2245     \exp_args:NNo
2246     \seq_put_right:No \l__stex_notation_precedences_seq {
2247       \l__stex_notation_opprec_tl
2248     }
2249   }
2250 }
2251
2252 \tl_clear:N \l__stex_notation_dummyargs_tl
2253
2254 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2255   \exp_args:NNe
2256   \cs_set:Npn \l__stex_notation_macrocode_cs {
2257     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2258     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2259     { \l__stex_notation_opprec_tl }
2260     { \exp_not:n { #2 } }
2261   }

```

```

2262   \__stex_notation_final:
2263 }{
2264   \str_if_in:NnTF \l__stex_notation_args_str b {
2265     \exp_args:Nne \use:nn
2266     {
2267       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2268       \cs_set:Npn \l__stex_notation_arity_str } { {
2269         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2270         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2271         { \l__stex_notation_opprec_tl }
2272         { \exp_not:n { #2 } }
2273       }}
2274 }{
2275   \str_if_in:NnTF \l__stex_notation_args_str B {
2276     \exp_args:Nne \use:nn
2277     {
2278       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2279       \cs_set:Npn \l__stex_notation_arity_str } { {
2280         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2281         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2282         { \l__stex_notation_opprec_tl }
2283         { \exp_not:n { #2 } }
2284       } }
2285 }{
2286   \exp_args:Nne \use:nn
2287   {
2288     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2289     \cs_set:Npn \l__stex_notation_arity_str } { {
2290       \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2291       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2292       { \l__stex_notation_opprec_tl }
2293       { \exp_not:n { #2 } }
2294     } }
2295   }
2296 }
2297
2298 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2299 \int_zero:N \l__stex_notation_currarg_int
2300 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2301 \__stex_notation_arguments:
2302 }
2303 }

```

(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

```

2304 \cs_new_protected:Nn \__stex_notation_arguments: {
2305   \int_incr:N \l__stex_notation_currarg_int
2306   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2307     \__stex_notation_final:
2308   }{
2309     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2310     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2311     \str_if_eq:VnTF \l_tmpa_str a {

```

```

2312     \_stex_notation_argument_assoc:n
2313   }{
2314     \str_if_eq:VnTF \l_tmpa_str B {
2315       \_stex_notation_argument_assoc:n
2316     }{
2317       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2318       \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2319         { \_stex_term_math_arg:nnn
2320           { \int_use:N \l__stex_notation_currarg_int }
2321           { \l_tmpa_str }
2322           { #####\int_use:N \l__stex_notation_currarg_int }
2323         }
2324       }
2325     } \_stex_notation_arguments:
2326   }
2327 }
2328 }
2329 }

```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2330 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2331
2332   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2333     {\l__stex_notation_arity_str}{
2334       #1
2335     }
2336   \int_zero:N \l_tmpa_int
2337   \tl_clear:N \l_tmpa_tl
2338   \str_map_inline:Nn \l__stex_notation_args_str {
2339     \int_incr:N \l_tmpa_int
2340     \tl_put_right:Nx \l_tmpa_tl {
2341       \str_if_eq:nnTF {##1}{a}{ {} }{
2342         \str_if_eq:nnTF {##1}{B}{ {} }{
2343           {##### \int_use:N \l_tmpa_int}
2344         }
2345       }
2346     }
2347   }
2348   \exp_after:wN\exp_after:wN\exp_after:wN \def
2349   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2350   \exp_after:wN\exp_after:wN\exp_after:wN ##
2351   \exp_after:wN\exp_after:wN\exp_after:wN 1
2352   \exp_after:wN\exp_after:wN\exp_after:wN ##
2353   \exp_after:wN\exp_after:wN\exp_after:wN 2
2354   \exp_after:wN\exp_after:wN\exp_after:wN {
2355     \exp_after:wN \exp_after:wN \exp_after:wN
2356     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2357       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2358     }
2359   }
2360
2361   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str

```

```

2362 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2363   \stex_term_math_assoc_arg:nnnn
2364   { \int_use:N \l__stex_notation_currarg_int }
2365   { \l_tmpa_str }
2366   { ####\int_use:N \l__stex_notation_currarg_int }
2367   { \l_tmpa_cs {####1} {####2} }
2368 } }
2369 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2370 %\tl_put_right:Nx \l_tmpa_tl {
2371 % { \stex_term_math_assoc_arg:nnnn
2372 % { \int_use:N \l_tmpa_int }
2373 % { \l_tmpb_str }
2374 % \exp_args:No \exp_not:n
2375 % {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2376 % { ####\int_use:N \l_tmpa_int }
2377 % }
2378 %}
2379 \__stex_notation_arguments:
2380 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2381 \cs_new_protected:Nn \__stex_notation_final: {
2382   \exp_args:Nne \use:nn
2383   {
2384     \cs_generate_from_arg_count:cNnn {
2385       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2386       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2387       _cs
2388     }
2389     \cs_set:Npn \l__stex_notation_arity_str { { {
2390       \exp_after:wN \exp_after:wN \exp_after:wN
2391       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2392       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2393     } }
2394
2395     \tl_if_empty:NF \l__stex_notation_op_tl {
2396       \cs_set:cpx {
2397         stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2398         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2399         _cs
2400       } {
2401         \stex_term_oms:nnn {
2402           \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2403           \l__stex_notation_lang_str
2404         }{
2405           \l__stex_notation_symbol_str
2406         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2407       }
2408     }
2409
2410     \exp_args:Ne
2411     \stex_add_to_current_module:n {

```

```

2412 \cs_generate_from_arg_count:cNnn {
2413   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2414   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2415   _cs
2416 } \cs_set:Npn {\l__stex_notation_arity_str} {
2417   \exp_after:wN \exp_after:wN \exp_after:wN
2418   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2419   { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2420 }
2421 \tl_if_empty:NF \l__stex_notation_op_tl {
2422   \cs_set:cpn {
2423     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2424     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2425     _cs
2426   } {
2427     \stex_term_oms:nnn {
2428       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2429       \l__stex_notation_lang_str
2430     }{
2431       \l__stex_notation_symbol_str
2432     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2433   }
2434 }
2435 }
2436
2437 \seq_put_right:cx {
2438   l_stex_symdecl_ \l__stex_notation_symbol_str
2439   _notations
2440 } {
2441   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2442 }
2443
2444 \stex_debug:nn{symbols}{
2445   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2446   ~for~\l__stex_notation_symbol_str^^J
2447   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2448   Argument~precedences:~
2449   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2450   Notation: \cs_meaning:c {
2451     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2452     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2453     _cs
2454   }
2455 }
2456
2457 %\prop_set_eq:cN {
2458 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2459 %   \c_hash_str \l__stex_notation_lang_str _prop
2460 %} \l_tmpb_prop
2461
2462 \exp_args:Ne
2463 \stex_add_to_current_module:n {
2464   \seq_put_right:cn {
2465     l_stex_symdecl_

```

```

2466         \prop_item:Nn \l_tmpb_prop { symbol }
2467     _notations
2468 } {
2469     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2470 }
2471 %\prop_set_from_keyval:cn {
2472 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2473 %   \c_hash_str \l__stex_notation_lang_str _prop
2474 %} {
2475 %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2476 %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2477 %   variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,
2478 %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2479 %   argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2480 %}
2481 }
2482
2483 \stex_if_smsmode:TF {
2484 %   \exp_args:Nx \stex_add_to_sms:n {
2485 %       \prop_set_from_keyval:cn {
2486 %           l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2487 %           \c_hash_str \l__stex_notation_lang_str _prop
2488 %       } {
2489 %           symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2490 %           language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2491 %           variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,
2492 %           opprec      = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2493 %           argprec     = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2494 %       }
2495 %   }
2496 }{
2497
2498 % HTML annotations
2499 \stex_if_do_html:T {
2500     \stex_annotate_invisible:nnn { notation }
2501     { \l__stex_notation_symbol_str } {
2502         \stex_annotate_invisible:nnn { notationfragment }
2503         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2504     }
2505     \stex_annotate_invisible:nnn { precedence }
2506     { \l__stex_notation_prec_str }{}
2507
2508     \int_zero:N \l_tmpa_int
2509     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2510     \tl_clear:N \l_tmpa_tl
2511     \int_step_inline:nn { \l__stex_notation_arity_str }{
2512         \int_incr:N \l_tmpa_int
2513         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2514         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2515         \str_if_eq:VnTF \l_tmpb_str a {
2516             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2517                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2518                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2519             } }
2520         }
2521     }
2522 }{

```



```

2520     \str_if_eq:VnTF \l_tmpb_str B {
2521       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2522         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2523         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2524       } }
2525     }{
2526       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2527         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2528       } }
2529     }
2530   }
2531 }
2532 \stex_annotate_invisible:nnn { notationcomp }{}{
2533   \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2534   $ \exp_args:Nno \use:nn { \use:c {
2535     stex_notation_ \l_stex_current_symbol_str
2536     \c_hash_str \l__stex_notation_variant_str
2537     \c_hash_str \l__stex_notation_lang_str _cs
2538   } } { \l_tmpa_tl } $
2539 }
2540 }
2541 }
2542 }
2543 \stex_smsmode_do:
2544 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

2545 \keys_define:nn { stex / setnotation } {
2546   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2547   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2548   unknown .code:n = \str_set:Nx
2549     \l__stex_notation_variant_str \l_keys_key_str
2550 }
2551
2552 \cs_new_protected:Nn \stex_setnotation_args:n {
2553   \str_clear:N \l__stex_notation_lang_str
2554   \str_clear:N \l__stex_notation_variant_str
2555   \keys_set:nn { stex / setnotation } { #1 }
2556 }
2557
2558 \NewDocumentCommand \setnotation {m m} {
2559   \stex_get_symbol:n { #1 }
2560   \stex_setnotation_args:n { #2 }
2561   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2562     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2563     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2564       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2565     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2566       { \c_hash_str }
2567     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2568       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2569     \exp_args:Nx \stex_add_to_current_module:n {

```

```

2570 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2571 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2572 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2573 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2574 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2575 { \c_hash_str }
2576 }
2577 \stex_debug:nn {notations}{
2578 Setting~default~notation~
2579 {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2580 \l_stex_get_symbol_uri_str \
2581 \expandafter\meaning\csname
2582 l_stex_symdecl_\l_stex_get_symbol_uri_str _notations\endcsname
2583 }
2584 }{
2585 % todo throw error
2586 }
2587 }
2588

```

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

\symdef

```

2589 \keys_define:nn { stex / symdef } {
2590   name .str_set_x:N = \l_stex_symdecl_name_str ,
2591   local .bool_set:N = \l_stex_symdecl_local_bool ,
2592   args .str_set_x:N = \l_stex_symdecl_args_str ,
2593   type .tl_set:N = \l_stex_symdecl_type_tl ,
2594   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2595   op .tl_set:N = \l__stex_notation_op_tl ,
2596   lang .str_set_x:N = \l__stex_notation_lang_str ,
2597   variant .str_set_x:N = \l__stex_notation_variant_str ,
2598   prec .str_set_x:N = \l__stex_notation_prec_str ,
2599   unknown .code:n = \str_set:Nx
2600     \l__stex_notation_variant_str \l_keys_key_str
2601 }
2602
2603 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2604   \str_clear:N \l_stex_symdecl_name_str
2605   \str_clear:N \l_stex_symdecl_args_str
2606   \bool_set_false:N \l_stex_symdecl_local_bool
2607   \tl_clear:N \l_stex_symdecl_type_tl
2608   \tl_clear:N \l_stex_symdecl_definiens_tl
2609   \str_clear:N \l__stex_notation_lang_str
2610   \str_clear:N \l__stex_notation_variant_str
2611   \str_clear:N \l__stex_notation_prec_str
2612   \tl_clear:N \l__stex_notation_op_tl
2613
2614   \keys_set:nn { stex / symdef } { #1 }
2615 }
2616
2617 \NewDocumentCommand \symdef { O{} m } {
2618   \__stex_notation_symdef_args:n { #1 }
2619   \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

2620 \stex_symdecl_do:n { #2 }
2621 \exp_args:Nx \stex_notation_do:nn {
2622   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2623 }
2624 }
2625 \stex_deactivate_macro:Nn \symdef {module~environments}

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

2626 \endpackage

```

Chapter 31

STEX -Terms Implementation

```
2627 <*package>
2628
2629 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2630
2631 <@@=stex_terms>
2632
2633   Warnings and error messages
2634   \msg_new:nnn{stex}{error/nonotation}{
2635     Symbol~#1~invoked,~but~has~no~notation~#2!
2636   }
2637   \msg_new:nnn{stex}{error/notationarg}{
2638     Error~in~parsing~notation~#1
2639   }
2640   \msg_new:nnn{stex}{error/noop}{
2641     Symbol~#1~has~no~operator~notation~for~notation~#2
2642   }
2643
```

31.1 Symbol Invocations

Arguments:

```
2642 \keys_define:nn { stex / terms } {
2643   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2644   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2645   unknown .code:n = \str_set:Nx
2646     \l__stex_terms_variant_str \l_keys_key_str
2647 }
2648
2649 \cs_new_protected:Nn \__stex_terms_args:n {
2650   \str_clear:N \l__stex_terms_lang_str
2651   \str_clear:N \l__stex_terms_variant_str
2652   \str_clear:N \l__stex_terms_prec_str
2653   \tl_clear:N \l__stex_terms_op_tl
2654
2655   \keys_set:nn { stex / terms } { #1 }
```

2656 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2657 \cs_new_protected:Nn \stex_invoke_symbol:n {
2658   \if_mode_math:
2659     \exp_after:wN \__stex_terms_invoke_math:n
2660   \else:
2661     \exp_after:wN \__stex_terms_invoke_text:n
2662   \fi: { #1 }
2663 }
```

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

__stex_terms_invoke_math:n

```
2664 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2665   \peek_charcode_remove:NTF ! {
2666     \peek_charcode:NTF [ {
2667       \__stex_terms_invoke_op:nw { #1 }
2668     }{
2669       \peek_charcode_remove:NTF ! {
2670         \peek_charcode:NTF [ {
2671           \__stex_terms_invoke_op_custom:nw
2672         }{
2673           % TODO throw error
2674         }
2675       }{
2676         \__stex_terms_invoke_op:nw { #1 } []
2677       }
2678     }{
2679       \peek_charcode_remove:NTF * {
2680         \__stex_terms_invoke_text:n { #1 }
2681       }{
2682         \peek_charcode:NTF [ {
2683           \__stex_terms_invoke_math:nw { #1 }
2684         }{
2685           \__stex_terms_invoke_math:nw { #1 } []
2686         }
2687       }
2688     }
2689   }
2690 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2691 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2692   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2693     \stex_highlight_term:nn{#1}{#2}
2694   }
2695 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2696 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2697   \_stex_terms_args:n { #2 }
2698   \cs_if_exist:cTF {
2699     stex_op_notation_ #1 \c_hash_str
2700     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2701   }{
2702     \csname stex_op_notation_ #1 \c_hash_str
2703       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2704     \endcsname
2705   }{
2706     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2707   }
2708 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2709 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2710   \_stex_terms_args:n { #2 }
2711   \seq_if_empty:cTF {
2712     l_stex_symdecl_ #1 _notations
2713   } {
2714     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2715   } {
2716     \seq_if_in:cxTF {
2717       l_stex_symdecl_ #1 _notations
2718     }
2719     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2720       \str_set:Nn \l_stex_current_symbol_str { #1 }
2721       \stex_debug:nn{terms}{Using~
2722         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \\
2723         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2724         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2725         _cs\endcsname
2726       }
2727       \use:c{
2728         stex_notation_ #1 \c_hash_str
2729         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2730         _cs
2731       }
2732     }{
2733       \str_if_empty:NTF \l__stex_terms_variant_str {
2734         \str_if_empty:NTF \l__stex_terms_lang_str {
2735           \seq_get_left:cN {
2736             l_stex_symdecl_ #1 _notations
2737           } \l_tmpa_str
2738           \str_set:Nn \l_stex_current_symbol_str { #1 }
2739           \stex_debug:nn{terms}{Using~
2740             #1\c_hash_str\l_tmpa_str \\
2741             \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2742             \l_tmpa_str
2743             _cs\endcsname
2744           }

```

```

2745         \use:c{
2746             stex_notation_ #1 \c_hash_str \l_tmpa_str
2747             _cs
2748         }
2749     }{
2750         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2751             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2752         }
2753     }
2754 }{
2755     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2756         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2757     }
2758 }
2759 }
2760 }
2761 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2762 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2763     \peek_charcode_remove:NTF ! {
2764         \stex_term_custom:nn { #1 } { }
2765     }{
2766         \prop_set_eq:Nc \l_tmpa_prop {
2767             l_stex_symdecl_ #1 _prop
2768         }
2769         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2770         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2771     }
2772 }

```

(End definition for `__stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2773 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2774 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2775 \int_new:N \l__stex_terms_downprec
2776 \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
2777 \tl_set:Nn \l__stex_terms_left_bracket_str (
2778 \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

```
2779 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2780   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2781     \bool_set_false:N \l__stex_terms_brackets_done_bool
2782     #2
2783   } {
2784     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2785       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2786         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2787         \dobrackets { #2 }
2788       }
2789     }{ #2 }
2790   }
2791 }
```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2792 \bool_new:N \l__stex_terms_brackets_done_bool
2793 %\RequirePackage{scalerel}
2794 \cs_new_protected:Npn \dobrackets #1 {
2795   %\ThisStyle{\if D@m@switch
2796   %   \exp_args:Nnx \use:nn
2797   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2798   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2799   % \else
2800   \exp_args:Nnx \use:nn
2801   {
2802     \bool_set_true:N \l__stex_terms_brackets_done_bool
2803     \int_set:Nn \l__stex_terms_downprec \infprec
2804     \l__stex_terms_left_bracket_str
2805     #1
2806   }
2807   {
2808     \bool_set_false:N \l__stex_terms_brackets_done_bool
2809     \l__stex_terms_right_bracket_str
2810     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2811   }
2812   %\fi}
2813 }
```

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

`\withbrackets`

```
2814 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2815   \exp_args:Nnx \use:nn
2816   {
2817     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2818     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2819     #3
2820   }
2821   {
2822     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2823     {\l__stex_terms_left_bracket_str}
```



```

2824 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2825 {\l__stex_terms_right_bracket_str}
2826 }
2827 }

```

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

`\STEXinvisible`

```

2828 \cs_new_protected:Npn \STEXinvisible #1 {
2829 \stex_annotate_invisible:n { #1 }
2830 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2831 \cs_new_protected:Nn \_stex_term_oms:nnn {
2832 \stex_annotate:nnn{ OMID }{ #2 }{
2833 \stex_highlight_term:nn { #1 } { #3 }
2834 }
2835 }
2836
2837 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2838 \__stex_terms_maybe_brackets:nn { #3 }{
2839 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2840 }
2841 }

```

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

`_stex_term_math_oma:nnnn`

```

2842 \cs_new_protected:Nn \_stex_term_oma:nnn {
2843 \stex_annotate:nnn{ OMA }{ #2 }{
2844 \stex_highlight_term:nn { #1 } { #3 }
2845 }
2846 }
2847
2848 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2849 \__stex_terms_maybe_brackets:nn { #3 }{
2850 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2851 }
2852 }

```

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

`_stex_term_math_omb:nnnn`

```

2853 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2854 \stex_annotate:nnn{ OMBIND }{ #2 }{
2855 \stex_highlight_term:nn { #1 } { #3 }
2856 }
2857 }
2858
2859 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2860 \__stex_terms_maybe_brackets:nn { #3 }{
2861 \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

2862 }
2863 }

```

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

`_stex_term_math_arg:nnn`

```

2864 \cs_new_protected:Nn \_stex_term_arg:nn {
2865   \stex_unhighlight_term:n {
2866     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2867   }
2868 }
2869 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2870   \exp_args:Nnx \use:nn
2871     { \int_set:Nn \l__stex_terms_downprec { #2 }
2872       \_stex_term_arg:nn { #1 }{ #3 }
2873     }
2874     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2875     }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2876 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2877   % TODO sequences
2878   \clist_set:Nn \l_tmpa_clist{ #3 }
2879   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2880     \tl_set:Nn \l_tmpa_tl { #3 }
2881   }{
2882     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
2883     \clist_reverse:N \l_tmpa_clist
2884     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2885
2886     \clist_map_inline:Nn \l_tmpa_clist {
2887       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2888         \exp_args:Nno
2889           \l_tmpa_cs { ##1 } \l_tmpa_tl
2890       }
2891     }
2892   }
2893   \exp_args:Nnno
2894     \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2895 }

```

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

`\stex_term_custom:nn`

```

2896 \cs_new_protected:Nn \stex_term_custom:nn {
2897   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2898   \str_set:Nn \l_tmpa_str { #2 }
2899   \tl_clear:N \l_tmpa_tl
2900   \int_zero:N \l_tmpa_int
2901   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2902   \__stex_terms_custom_loop:
2903 }

```

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

__stex_terms_custom_loop:

```

2904 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2905   \bool_set_false:N \l_tmpa_bool
2906   \bool_while_do:nn {
2907     \str_if_eq_p:ee X {
2908       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2909     }
2910   }{
2911     \int_incr:N \l_tmpa_int
2912   }
2913
2914   \peek_charcode:NTF [ {
2915     % notation/text component
2916     \__stex_terms_custom_component:w
2917   } {
2918     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2919       % all arguments read => finish
2920       \__stex_terms_custom_final:
2921     } {
2922       % arguments missing
2923       \peek_charcode_remove:NTF * {
2924         % invisible, specific argument position or both
2925         \peek_charcode:NTF [ {
2926           % visible specific argument position
2927           \__stex_terms_custom_arg:wn
2928         } {
2929           % invisible
2930           \peek_charcode_remove:NTF * {
2931             % invisible specific argument position
2932             \__stex_terms_custom_arg_inv:wn
2933           } {
2934             % invisible next argument
2935             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2936           }
2937         }
2938       } {
2939         % next normal argument
2940         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2941       }
2942     }
2943   }
2944 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2945 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2946   \bool_set_true:N \l_tmpa_bool
2947   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2948 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2949 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
2950   \str_set:Nx \l_tmpb_str {
2951     \str_item:Nn \l_tmpa_str { #1 }
2952   }
2953   \str_case:VnTF \l_tmpb_str {
2954     { X } {
2955       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2956     }
2957     { i } { \_stex_terms_custom_set_X:n { #1 } }
2958     { b } { \_stex_terms_custom_set_X:n { #1 } }
2959     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2960     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2961   }{}{
2962     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2963   }
2964
2965   \bool_if:nTF \l_tmpa_bool {
2966     \tl_put_right:Nx \l_tmpa_tl {
2967       \stex_annotate_invisible:n {
2968         \stex_term_arg:nn { \int_eval:n { #1 } }
2969         \exp_not:n { { #2 } }
2970       }
2971     }
2972   } {
2973     \tl_put_right:Nx \l_tmpa_tl {
2974       \stex_term_arg:nn { \int_eval:n { #1 } }
2975       \exp_not:n { { #2 } }
2976     }
2977   }
2978
2979   \_stex_terms_custom_loop:
2980 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2981 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2982   \str_set:Nx \l_tmpa_str {
2983     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2984     X
2985     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2986   }
2987 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2988 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2989   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2990   \_stex_terms_custom_loop:
2991 }

```

(End definition for _stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

2992 \cs_new_protected:Nn \_stex_terms_custom_final: {
2993   \int_compare:nNnTF \l_tmpb_int = 0 {
2994     \exp_args:Nnno \_stex_term_oms:nnn
2995   }{
2996     \str_if_in:NnTF \l_tmpa_str {b} {
2997       \exp_args:Nnno \_stex_term_ombind:nnn
2998     } {
2999       \exp_args:Nnno \_stex_term_oma:nnn
3000     }
3001   }
3002   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3003 }

```

(End definition for _stex_terms_custom_final:.)

`\symref`
`\symname`

```

3004 \NewDocumentCommand \symref { m m }{
3005   \let\compemph_uri_prev:\compemph@uri
3006   \let\compemph@uri\symrefemph@uri
3007   \STEXsymbol{#1}![#2]
3008   \let\compemph@uri\compemph_uri_prev:
3009 }
3010
3011 \keys_define:nn { stex / symname } {
3012   post      .str_set_x:N      = \l_stex_symname_post_str
3013 }
3014
3015 \cs_new_protected:Nn \stex_symname_args:n {
3016   \str_clear:N \l_stex_symname_post_str
3017   \keys_set:nn { stex / symname } { #1 }
3018 }
3019
3020 \NewDocumentCommand \symname { 0{} m }{
3021   \stex_symname_args:n { #1 }
3022   \stex_get_symbol:n { #2 }
3023   \str_set:Nx \l_tmpa_str {
3024     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3025   }
3026   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3027
3028   \let\compemph_uri_prev:\compemph@uri
3029   \let\compemph@uri\symrefemph@uri
3030   \exp_args:NNx \use:nn
3031   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3032     \l_tmpa_str \l_stex_symname_post_str
3033   ] }
3034   \let\compemph@uri\compemph_uri_prev:
3035 }

```

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

31.3 Notation Components

3036 <@@=stex_notationcomps>

`\stex_highlight_term:nn`

```

3037
3038 \str_new:N \l_stex_current_symbol_str
3039 \cs_new_protected:Nn \stex_highlight_term:nn {
3040   \exp_args:Nnx
3041   \use:nn {
3042     \str_set:Nx \l_stex_current_symbol_str { #1 }
3043     #2
3044   } {
3045     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3046     { \l_stex_current_symbol_str }
3047   }
3048 }
3049
3050 \cs_new_protected:Nn \stex_unhighlight_term:n {
3051   % \latexml_if:TF {
3052   %   #1
3053   % } {
3054   %   \rustex_if:TF {
3055   %     #1
3056   %   } {
3057     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3058   %   }
3059   % }
3060 }
```

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

```

\comp
\compemph@uri 3061 \cs_new_protected:Npn \comp #1 {
\compemph 3062   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3063     \rustex_if:TF {
\defemph@uri 3064       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3065     }{
\symrefemph@uri 3066       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3067     }
3068   }
3069 }
3070
3071 \cs_new_protected:Npn \compemph@uri #1 #2 {
3072   \compemph{ #1 }
3073 }
3074
3075
3076 \cs_new_protected:Npn \compemph #1 {
3077   #1
3078 }
3079
3080 \cs_new_protected:Npn \defemph@uri #1 #2 {
3081   \defemph{#1}
3082 }
```

```

3083
3084 \cs_new_protected:Npn \defemph #1 {
3085     \textbf{#1}
3086 }
3087
3088 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3089     \symrefemph{#1}
3090 }
3091
3092 \cs_new_protected:Npn \symrefemph #1 {
3093     \textbf{#1}
3094 }

```

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

`\ellipses`

```

3095 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3096 \bool_new:N \l_stex_inparray_bool
3097 \bool_set_false:N \l_stex_inparray_bool
3098 \NewDocumentCommand \parray { m m } {
3099     \begingroup
3100     \bool_set_true:N \l_stex_inparray_bool
3101     \begin{array}{#1}
3102         #2
3103     \end{array}
3104     \endgroup
3105 }
3106
3107 \NewDocumentCommand \prmatrix { m } {
3108     \begingroup
3109     \bool_set_true:N \l_stex_inparray_bool
3110     \begin{matrix}
3111         #1
3112     \end{matrix}
3113     \endgroup
3114 }
3115
3116 \def \maybepline {
3117     \bool_if:NT \l_stex_inparray_bool {\hline}
3118 }
3119
3120 \def \parrayline #1 #2 {
3121     #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
3122 }
3123
3124 \def \pmrow #1 { \parrayline{#1} }
3125
3126 \def \parraylineh #1 #2 {
3127     #1 #2 \bool_if:NT \l_stex_inparray_bool {\hline}
3128 }
3129

```

```

3130 \def \parraycell #1 {
3131   #1 \bool_if:NT \l_stex_inarray_bool {&}
3132 }

```

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

```

3133 \endpackage

```


Chapter 32

STEX -Structural Features Implementation

```
3134 <*package>
3135
3136 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3137
3138 <@@=stex_features>
3139
3140 Warnings and error messages
3141 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3142   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3143 }
3144 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3145   Symbol~#1~not~assigned~in~interpretmodule~#2
3146 }
3147
```

32.1 Imports with modification

```
3146 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3147   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3148     \__stex_features_get_symbol_from_cs:n { #1 }
3149   }{
3150     % argument is a string
3151     % is it a command name?
3152     \cs_if_exist:cTF { #1 }{
3153       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3154       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3155       \str_if_empty:NNTF \l_tmpa_str {
3156         \exp_args:Nx \cs_if_eq:NNTF {
3157           \tl_head:N \l_tmpa_tl
3158         } \stex_invoke_symbol:n {
3159           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3160         }{
3161           \__stex_features_get_symbol_from_string:n { #1 }
3162         }
3163       }
3164     }
3165   }
3166 }
```

```

3162     }
3163   } {
3164     \__stex_features_get_symbol_from_string:n { #1 }
3165   }
3166   }{
3167     % argument is not a command name
3168     \__stex_features_get_symbol_from_string:n { #1 }
3169     % \l_stex_all_symbols_seq
3170   }
3171 }
3172 }
3173
3174 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3175   \str_set:Nn \l_tmpa_str { #1 }
3176   \bool_set_false:N \l_tmpa_bool
3177   \bool_if:NF \l_tmpa_bool {
3178     \tl_set:Nn \l_tmpa_tl {
3179       \msg_set:nnn{stex}{error/unknownsymbol}{
3180         No~symbol~#1~found!
3181       }
3182       \msg_error:nn{stex}{error/unknownsymbol}
3183     }
3184     \str_set:Nn \l_tmpa_str { #1 }
3185     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3186     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3187       \str_set:Nn \l_tmpb_str { ##1 }
3188       \str_if_eq:eeT { \l_tmpa_str } {
3189         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3190       } {
3191         \seq_map_break:n {
3192           \tl_set:Nn \l_tmpa_tl {
3193             \str_set:Nn \l_stex_get_symbol_uri_str {
3194               ##1
3195             }
3196             \__stex_features_get_symbol_check:
3197           }
3198         }
3199       }
3200     }
3201     \l_tmpa_tl
3202   }
3203 }
3204
3205 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3206   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3207   { \tl_tail:N \l_tmpa_tl }
3208   \tl_if_single:NTF \l_tmpa_tl {
3209     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3210       \exp_after:wN \str_set:Nn \exp_after:wN
3211       \l_stex_get_symbol_uri_str \l_tmpa_tl
3212       \__stex_features_get_symbol_check:
3213     }{
3214       % TODO
3215       % tail is not a single group

```

```

3216     }
3217 }{
3218     % TODO
3219     % tail is not a single group
3220 }
3221 }
3222
3223 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3224     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3225     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3226         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3227         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3228         \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3229             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3230                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3231             }
3232         }
3233     }{
3234         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3235             \l_stex_current_copymodule_name_str~(inexplicably)
3236         }
3237     }
3238 }
3239
3240 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3241     \stex_import_module_uri:nn { #1 } { #2 }
3242     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3243     \stex_import_require_module:nnnn
3244     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3245     { \l_stex_import_path_str } { \l_stex_import_name_str }
3246     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3247     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3248     \seq_clear:N \l__stex_features_copymodule_fields_seq
3249     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3250         \seq_map_inline:cn {c_stex_module_###1_constants}{
3251             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3252                 ###1 ? #####1
3253             }
3254         }
3255     }
3256     \seq_clear:N \l_tmpa_seq
3257     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3258         name      = \l_stex_current_copymodule_name_str ,
3259         module     = \l_stex_current_module_str ,
3260         from       = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3261         includes   = \l_tmpa_seq ,
3262         fields     = \l_tmpa_seq
3263     }
3264     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3265         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3266     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3267     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3268     \stex_if_smsmode:F {
3269         \begin{stex_annotate_env} {#4} {

```

```

3270     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3271   }
3272   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3273 }
3274 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3275 \bool_set_false:N \l_stex_html_do_output_bool
3276 }
3277 \cs_new_protected:Nn \stex_copymodule_end:n {
3278   \def \l_tmpa_cs ##1 ##2 {#1}
3279   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3280   \tl_clear:N \l_tmpa_tl
3281   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3282   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3283     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3284       \l_tmpa_cs{##1}{####1}
3285       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3286         \tl_put_right:Nx \l_tmpa_tl {
3287           \prop_set_from_keyval:cn {
3288             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3289           }{
3290             \exp_after:wN \prop_to_keyval:N \csname
3291               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3292             \endcsname
3293           }
3294           \seq_clear:c {
3295             l_stex_symdecl_
3296             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3297             _notations
3298           }
3299         }
3300         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_na
3301         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3302         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3303           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3304           \tl_put_right:Nx \l_tmpa_tl {
3305             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3306             \stex_invoke_symbol:n {
3307               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3308             }
3309           }
3310         }
3311       }
3312     }{
3313       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3314       \prop_put:Nnx \l_tmpa_prop { name }{\l_stex_current_copymodule_name_str / ####1 }
3315       \prop_put:Nnx \l_tmpa_prop { module }{\l_stex_current_module_str }
3316       \tl_put_right:Nx \l_tmpa_tl {
3317         \prop_set_from_keyval:cn {
3318           l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3319         }{
3320           \prop_to_keyval:N \l_tmpa_prop
3321         }
3322         \seq_clear:c {
3323           l_stex_symdecl_

```

```

3324         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3325         _notations
3326     }
3327 }
3328 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1}
3329 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3330     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3331     \tl_put_right:Nx \l_tmpa_tl {
3332         \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3333             \stex_invoke_symbol:n {
3334                 \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3335             }
3336         }
3337     }
3338 }
3339 }
3340 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3341     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3342 }
3343 % todo notations
3344 }}
3345 }
3346 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3347 \tl_put_left:Nx \l_tmpa_tl {
3348     \prop_set_from_keyval:cn {
3349         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3350     }{
3351         \prop_to_keyval:N \l_stex_current_copymodule_prop
3352     }
3353 }
3354 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3355 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3356 \exp_args:Nx \stex_do_aftergroup:n {
3357     \exp_args:No \exp_not:n \l_tmpa_tl
3358 }
3359 \stex_if_smsmode:F {
3360     \end{stex_annotate_env}
3361 }
3362 }
3363
3364 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3365     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3366     \stex_deactivate_macro:Nn \symdecl {module~environments}
3367     \stex_deactivate_macro:Nn \symdef {module~environments}
3368     \stex_deactivate_macro:Nn \notation {module~environments}
3369     \stex_reactivate_macro:N \assign
3370     \stex_reactivate_macro:N \renamedec1
3371     \stex_reactivate_macro:N \donotcopy
3372     \stex_smsmode_do:
3373 }{
3374     \stex_copymodule_end:n {}
3375 }
3376
3377 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{

```

```

3378 \stex_copymodule_start:nnnn { #1 } { #2 } { #3 } { realization }
3379 \stex_deactivate_macro:Nn \symdecl {module~environments}
3380 \stex_deactivate_macro:Nn \symdef {module~environments}
3381 \stex_deactivate_macro:Nn \notation {module~environments}
3382 \stex_reactivate_macro:N \assign
3383 \stex_reactivate_macro:N \renamedekl
3384 \stex_reactivate_macro:N \donotcopy
3385 \stex_smsmode_do:
3386 }{
3387 \stex_copymodule_end:n {
3388 \tl_if_exist:cF {
3389 l__stex_features_copymodule_##1?##2_def_tl
3390 }{
3391 \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3392 ##1?##2
3393 }{\l_stex_current_copymodule_name_str}
3394 }
3395 }
3396 }
3397
3398 \NewDocumentCommand \donotcopy { 0{} m }{
3399 \stex_import_module_uri:nn { #1 } { #2 }
3400 \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3401 \seq_map_inline:Nn \l_stex_collect_imports_seq {
3402 \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3403 \seq_map_inline:cn {c_stex_module_##1_constants}{
3404 \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3405 \bool_lazy_any_p:nT {
3406 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3407 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3408 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3409 }{
3410 % TODO throw error
3411 }
3412 }
3413 }
3414
3415 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3416 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3417 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3418 }
3419
3420 \NewDocumentCommand \assign { m m }{
3421 \stex_get_symbol_in_copymodule:n {#1}
3422 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3423 \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3424 }
3425
3426 \keys_define:nn { stex / renamedekl } {
3427 name .str_set_x:N = \l_stex_renamedekl_name_str
3428 }
3429 \cs_new_protected:Nn \__stex_features_renamedekl_args:n {
3430 \str_clear:N \l_stex_renamedekl_name_str
3431

```

```

3432 \keys_set:nn { stex / renamedecl } { #1 }
3433 }
3434
3435 \NewDocumentCommand \renamedecl { 0{} m m}{
3436 \__stex_features_renamedecl_args:n { #1 }
3437 \stex_get_symbol_in_copymodule:n {#2}
3438 \stex_debug:nn{renamedecl}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3439 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3440 \str_if_empty:NTF \l_stex_renamedecl_name_str {
3441 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3442 \l_stex_get_symbol_uri_str
3443 } }
3444 } {
3445 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3446 \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
3447 \prop_set_eq:cc {l_stex_symdecl_
3448 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3449 _prop
3450 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3451 \seq_set_eq:cc {l_stex_symdecl_
3452 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3453 _notations
3454 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3455 \prop_put:cnx {l_stex_symdecl_
3456 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3457 _prop
3458 }{ name }{ \l_stex_renamedecl_name_str }
3459 \prop_put:cnx {l_stex_symdecl_
3460 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3461 _prop
3462 }{ module }{ \l_stex_current_module_str }
3463 \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3464 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3465 }
3466 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3467 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
3468 } }
3469 }
3470 }
3471 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3472 % \_stex_notation_args:n { #1 }
3473 % \tl_clear:N \l_stex_symdecl_definiens_tl
3474 % \stex_get_symbol_in_copymodule:n { #2 }
3475 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3476 % % todo
3477 %}
3478 \stex_deactivate_macro:Nn \assign {copymodules}
3479 \stex_deactivate_macro:Nn \renamedecl {copymodules}
3480 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3481
3482
3483 \seq_new:N \l_stex_implicit_morphisms_seq
3484 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3485 \stex_import_module_uri:nn { #1 } { #2 }

```

```

3486 \stex_debug:nn{implicits}{
3487   Implicit~morphism:~
3488   \l_stex_module_ns_str ? \l__stex_features_name_str
3489 }
3490 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3491   \l_stex_module_ns_str ? \l__stex_features_name_str
3492 }{
3493   \msg_error:nnn{stex}{error/conflictingmodules}{
3494     \l_stex_module_ns_str ? \l__stex_features_name_str
3495   }
3496 }
3497
3498 % TODO
3499
3500
3501
3502 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3503   \l_stex_module_ns_str ? \l__stex_features_name_str
3504 }
3505 }
3506

```

32.2 The feature environment

structural@feature

```

3507
3508 \NewDocumentEnvironment{structural@feature}{ m m m }{
3509   \stex_if_in_module:F {
3510     \msg_set:nnn{stex}{error/nomodule}{
3511       Structural~Feature~has~to~occur~in~a~module:\\
3512       Feature~#2~of~type~#1\\
3513       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3514     }
3515     \msg_error:nn{stex}{error/nomodule}
3516   }
3517
3518   \str_set:Nx \l_stex_module_name_str {
3519     \prop_item:Nn \l_stex_current_module_prop
3520     { name } / #2 - feature
3521   }
3522
3523   \str_set:Nx \l_stex_module_ns_str {
3524     \prop_item:Nn \l_stex_current_module_prop
3525     { ns }
3526   }
3527
3528
3529   \str_clear:N \l_tmpa_str
3530   \seq_clear:N \l_tmpa_seq
3531   \tl_clear:N \l_tmpa_tl
3532   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3533     origname = #2,
3534     name      = \l_stex_module_name_str ,
3535     ns        = \l_stex_module_ns_str ,

```



```

3536 imports = \exp_not:o { \l_tmpa_seq } ,
3537 constants = \exp_not:o { \l_tmpa_seq } ,
3538 content = \exp_not:o { \l_tmpa_tl } ,
3539 file = \exp_not:o { \g_stex_currentfile_seq } ,
3540 lang = \l_stex_module_lang_str ,
3541 sig = \l_tmpa_str ,
3542 meta = \l_tmpa_str ,
3543 feature = #1 ,
3544 }
3545
3546 \stex_if_smsmode:F {
3547   \begin{stex_annotate_env}{ feature:#1 }{}
3548   \stex_annotate_invisible:nnn{header}{}{ #3 }
3549 }
3550 {}
3551 \str_set:Nx \l_tmpa_str {
3552   c_stex_feature_
3553   \prop_item:Nn \l_stex_current_module_prop { ns } ?
3554   \prop_item:Nn \l_stex_current_module_prop { name }
3555   _prop
3556 }
3557 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3558 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3559 \stex_if_smsmode:TF {
3560   \exp_args:Nx \stex_add_to_sms:n {
3561     \prop_gset_from_keyval:cn {
3562       c_stex_feature_
3563       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3564       \prop_item:Nn \l_stex_current_module_prop { name }
3565       _prop
3566     } {
3567       origname = #2,
3568       name = \prop_item:cn { \l_tmpa_str } { name } ,
3569       ns = \prop_item:cn { \l_tmpa_str } { ns } ,
3570       imports = \prop_item:cn { \l_tmpa_str } { imports } ,
3571       constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3572       content = \prop_item:cn { \l_tmpa_str } { content } ,
3573       file = \prop_item:cn { \l_tmpa_str } { file } ,
3574       lang = \prop_item:cn { \l_tmpa_str } { lang } ,
3575       sig = \prop_item:cn { \l_tmpa_str } { sig } ,
3576       meta = \prop_item:cn { \l_tmpa_str } { meta } ,
3577       feature = \prop_item:cn { \l_tmpa_str } { feature }
3578     }
3579   }
3580 } {
3581   \end{stex_annotate_env}
3582 }
3583 }
3584

```

32.3 Features

structure

```

3585
3586 \prop_new:N \l_stex_all_structures_prop
3587
3588 \keys_define:nn { stex / features / structure } {
3589   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3590 }
3591
3592 \cs_new_protected:Nn \__stex_features_structure_args:n {
3593   \str_clear:N \l__stex_features_structure_name_str
3594   \keys_set:nn { stex / features / structure } { #1 }
3595 }
3596
3597 %\stex_new_feature:nnnn { structure } { 0{} m } {
3598 %   \__stex_features_structure_args:n { ##1 }
3599 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3600 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3601 %   }
3602 %} {
3603 %
3604 %}
3605
3606 \NewDocumentEnvironment{mathstructure}{0{} m }{
3607   \__stex_features_structure_args:n { #1 }
3608   \str_if_empty:NT \l__stex_features_structure_name_str {
3609     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3610   }
3611   \exp_args:Nnnx
3612   \begin{structural@feature}{ structure }
3613     { \l__stex_features_structure_name_str }{}
3614     \seq_clear:N \l_tmpa_seq
3615     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3616     \stex_smsmode_do:
3617   }{
3618     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3619     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3620     \str_set:Nx \l_tmpa_str {
3621       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3622       \prop_item:Nn \l_stex_current_module_prop { name }
3623     }
3624     \seq_map_inline:Nn \l_tmpa_seq {
3625       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3626     }
3627     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3628     \exp_args:Nnx
3629     \AddToHookNext { env / mathstructure / after }{
3630       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3631         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3632       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3633     \STEXexport {
3634       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3635       { \prop_item:Nn \l_stex_current_module_prop { origname } }
3636       { \l_tmpa_str }
3637       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3638       { #2 }{ \l_tmpa_str }

```

```

3639 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3640 %      \prop_item:Nn \l_stex_current_module_prop { origname },
3641 %      \l_tmpa_str
3642 %      }
3643 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3644 %      #2,\l_tmpa_str
3645 %      }
3646 %      \tl_set:cx { #2 } {
3647 %      \stex_invoke_structure:n { \l_tmpa_str }
3648 %      }
3649 %      }
3650
3651 \end{structural@feature}
3652 % \g_stex_last_feature_prop
3653 }

```

\instantiate

```

3654 \seq_new:N \l__stex_features_structure_field_seq
3655 \str_new:N \l__stex_features_structure_field_str
3656 \str_new:N \l__stex_features_structure_def_tl
3657 \prop_new:N \l__stex_features_structure_prop
3658 \NewDocumentCommand \instantiate { m O{} m }{
3659   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3660   \prop_set_eq:Nc \l__stex_features_structure_prop {
3661     c_stex_feature_\l_tmpa_str _prop
3662   }
3663   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3664   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3665     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3666     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3667       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3668       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3669         {!} \l_tmpa_tl
3670       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3671         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3672         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3673         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3674       }{
3675         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3676         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3677         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3678           \l_tmpa_tl
3679         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3680           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3681           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3682         }{
3683           \tl_clear:N \l_tmpb_tl
3684         }
3685       }
3686     }{
3687       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3688       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3689         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3690         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl

```

```

3691         \tl_clear:N \l_tmpa_tl
3692     }{
3693         % TODO throw error
3694     }
3695 }
3696 % \l_tmpa_str: name
3697 % \l_tmpa_tl: definiens
3698 % \l_tmpb_tl: notation
3699 \tl_if_empty:NT \l__stex_features_structure_field_str {
3700     % TODO throw error
3701 }
3702 \str_clear:N \l_tmpb_str
3703
3704 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3705 \seq_map_inline:Nn \l_tmpa_seq {
3706     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3707     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3708     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3709         \seq_map_break:n {
3710             \str_set:Nn \l_tmpb_str { ####1 }
3711         }
3712     }
3713 }
3714 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3715     \l_tmpb_str
3716
3717 \tl_if_empty:NTF \l_tmpb_tl {
3718     \tl_if_empty:NF \l_tmpa_tl {
3719         \exp_args:Nx \use:n {
3720             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3721         }
3722     }
3723 }{
3724     \tl_if_empty:NTF \l_tmpa_tl {
3725         \exp_args:Nx \use:n {
3726             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3727         }
3728     }
3729 }{
3730     \exp_args:Nx \use:n {
3731         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3732         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3733     }
3734 }
3735 }
3736 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3737 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3738 % #3/\l__stex_features_structure_field_str
3739 % \par
3740 % \expandafter\present\csname
3741 %     l_stex_symdecl_
3742 %     \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 %      _prop
3746 %      \endcsname
3747 }
3748
3749 \tl_clear:N \l__stex_features_structure_def_tl
3750
3751 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3752 \seq_map_inline:Nn \l_tmpa_seq {
3753   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3754   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3755   \exp_args:Nx \use:n {
3756     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3757
3758     }
3759   }
3760
3761   \prop_if_exist:cF {
3762     l_stex_symdecl_
3763     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3764     \prop_item:Nn \l_stex_current_module_prop {name} ?
3765     #3/\l_tmpa_str
3766     _prop
3767   }{
3768     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3769     \l_tmpb_str
3770     \exp_args:Nx \use:n {
3771       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3772     }
3773   }
3774 }
3775
3776 \symdecl*[type={\STEXsymbol{module-type}}{
3777   \_stex_term_math_oms:nnnn {
3778     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3779     \prop_item:Nn \l__stex_features_structure_prop {name}
3780     }{0}{0}
3781   }]{#3}
3782
3783 % TODO: -> sms file
3784
3785 \tl_set:cx{ #3 }{
3786   \stex_invoke_structure:nnn {
3787     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3788     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3789   } {
3790     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3791     \prop_item:Nn \l__stex_features_structure_prop {name}
3792   }
3793 }
3794 \stex_smsmode_do:
3795 }

```

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

\stex_invoke_structure:nnn

```

3796 % #1: URI of the instance
3797 % #2: URI of the instantiated module
3798 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3799   \tl_if_empty:nTF{ #3 }{
3800     \prop_set_eq:Nc \l__stex_features_structure_prop {
3801       c_stex_feature_ #2 _prop
3802     }
3803     \tl_clear:N \l_tmpa_tl
3804     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3805     \seq_map_inline:Nn \l_tmpa_seq {
3806       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3807       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3808       \cs_if_exist:cT {
3809         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3810       }{
3811         \tl_if_empty:NF \l_tmpa_tl {
3812           \tl_put_right:Nn \l_tmpa_tl {,}
3813         }
3814         \tl_put_right:Nx \l_tmpa_tl {
3815           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3816         }
3817       }
3818     }
3819     \exp_args:No \mathstruct \l_tmpa_tl
3820   }{
3821     \stex_invoke_symbol:n{#1/#3}
3822   }
3823 }

```

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

3824 </package>

Chapter 33

STEX -Statements Implementation

```
3825 <*package>
3826
3827 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3828
3829 \protected\def\ignorespacesandpars{
3830   \begingroup\catcode13=10\relax
3831   \@ifnextchar\par{
3832     \endgroup\expandafter\ignorespacesandpars\@gobble
3833   }{
3834     \endgroup
3835   }
3836 }
3837
3838 <@@=stex_statements>
3839
3840 Warnings and error messages
```

\titleemph

```
3840 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3841 \keys_define:nn {stex / definiendum }{
3842   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3843   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3844   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3845 }
3846 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3847   \str_clear:N \l__stex_statements_definiendum_root_str
3848   \tl_clear:N \l__stex_statements_definiendum_post_tl
3849   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3850 \keys_set:nn { stex / definiendum } { #1 }
3851 }
3852 \NewDocumentCommand \definiendum { 0{} m m } {
3853   \__stex_statements_definiendum_args:n { #1 }
3854   \stex_get_symbol:n { #2 }
3855   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3856   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3857     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3858       \tl_set:Nn \l_tmpa_tl { #3 }
3859     } {
3860       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3861       \tl_set:Nn \l_tmpa_tl {
3862         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3863       }
3864     }
3865   } {
3866     \tl_set:Nn \l_tmpa_tl { #3 }
3867   }
3868
3869   % TODO root
3870   \rustex_if:TF {
3871     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3872   } {
3873     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3874   }
3875 }
3876 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3877
3878 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3879
3880 \NewDocumentCommand \definame { 0{} m } {
3881   \__stex_statements_definiendum_args:n { #1 }
3882   % TODO: root
3883   \stex_get_symbol:n { #2 }
3884   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3885   \str_set:Nx \l_tmpa_str {
3886     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3887   }
3888   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3889   \rustex_if:TF {
3890     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3891       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3892     }
3893   } {
3894     \defemph@uri {
3895       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3896     } { \l_stex_get_symbol_uri_str }
3897   }
3898 }
3899 \stex_deactivate_macro:Nn \definame {definition~environments}

```



```

3900
3901 \NewDocumentCommand \Definame { 0{ } m } {
3902   \_stex_statements_definiendum_args:n { #1 }
3903   \stex_get_symbol:n { #2 }
3904   \str_set:Nx \l_tmpa_str {
3905     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3906   }
3907   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3908   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3909   \rustex_if:TF {
3910     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3911       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3912     }
3913   } {
3914     \defemph@uri {
3915       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3916     } { \l_stex_get_symbol_uri_str }
3917   }
3918 }
3919 \stex_deactivate_macro:Nn \Definame {definition-environments}
3920
3921 \NewDocumentCommand \Symname { 0{ } m }{
3922   \stex_symname_args:n { #1 }
3923   \stex_get_symbol:n { #2 }
3924   \str_set:Nx \l_tmpa_str {
3925     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3926   }
3927   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3928   \let\compemph_uri_prev:\compemph@uri
3929   \let\compemph@uri\symrefemph@uri
3930   \exp_args:NNx \use:nn
3931   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3932     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3933     \l_stex_symname_post_str
3934   ] }
3935   \let\compemph@uri\compemph_uri_prev:
3936 }

```

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

sdefinition

```

3937
3938 \keys_define:nn {stex / sdefinition }{
3939   type      .str_set_x:N = \sdefinitiontype,
3940   id        .str_set_x:N = \sdefinitionid,
3941   name      .str_set_x:N = \sdefinitionname,
3942   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3943   title     .tl_set:N     = \sdefinitiontitle
3944 }
3945 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
3946   \str_clear:N \sdefinitiontype
3947   \str_clear:N \sdefinitionid
3948   \str_clear:N \sdefinitionname
3949   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

3950 \tl_clear:N \sdefinitiontitle
3951 \keys_set:nn { stex / sdefinition }{ #1 }
3952 }
3953
3954 \NewDocumentEnvironment{sdefinition}{O{}}{
3955   \__stex_statements_sdefinition_args:n{ #1 }
3956   \stex_reactivate_macro:N \definiendum
3957   \stex_reactivate_macro:N \definame
3958   \stex_reactivate_macro:N \Definame
3959   \stex_if_smsmode:F{
3960     \seq_clear:N \l_tmpa_seq
3961     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
3962       \str_if_eq:nnF{ ##1 }{ }{
3963         \stex_get_symbol:n { ##1 }
3964         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3965           \l_stex_get_symbol_uri_str
3966         }
3967       }
3968     }
3969     \exp_args:Nnnx
3970     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
3971     \str_if_empty:NF \sdefinitiontype {
3972       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
3973     }
3974     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
3975     \tl_clear:N \l_tmpa_tl
3976     \clist_map_inline:Nn \l_tmpa_clist {
3977       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3978         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3979       }
3980     }
3981     \tl_if_empty:NTF \l_tmpa_tl {
3982       \__stex_statements_sdefinition_start:
3983     }{
3984       \l_tmpa_tl
3985     }
3986   }
3987   \stex_ref_new_doc_target:n \sdefinitionid
3988   \stex_smsmode_do:
3989 }{
3990   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
3991   \stex_if_smsmode:F {
3992     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
3993     \tl_clear:N \l_tmpa_tl
3994     \clist_map_inline:Nn \l_tmpa_clist {
3995       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3996         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3997       }
3998     }
3999     \tl_if_empty:NTF \l_tmpa_tl {
4000       \__stex_statements_sdefinition_end:
4001     }{
4002       \l_tmpa_tl
4003     }

```

```

4004     \end{stex_annotate_env}
4005   }
4006 }

```

\stexpatchdefinition

```

4007 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4008   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4009     ~(\sdefinitiontitle)
4010   }~}
4011 }
4012 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4013
4014 \newcommand\stexpatchdefinition[3] [] {
4015   \str_set:Nx \l_tmpa_str{ #1 }
4016   \str_if_empty:NTF \l_tmpa_str {
4017     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4018     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4019   }{
4020     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4021     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4022   }
4023 }

```

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

\inlinedef inline:

```

4024 \keys_define:nn {stex / inlinedef }{
4025   type      .str_set_x:N = \sdefinitiontype,
4026   id        .str_set_x:N = \sdefinitionid,
4027   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4028   name      .str_set_x:N = \sdefinitionname
4029 }
4030 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4031   \str_clear:N \sdefinitiontype
4032   \str_clear:N \sdefinitionid
4033   \str_clear:N \sdefinitionname
4034   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4035   \keys_set:nn { stex / inlinedef }{ #1 }
4036 }
4037 \NewDocumentCommand \inlinedef { 0{} m } {
4038   \begingroup
4039   \__stex_statements_inlinedef_args:n{ #1 }
4040   \stex_ref_new_doc_target:n \sdefinitionid
4041   \stex_reactivate_macro:N \definiendum
4042   \stex_reactivate_macro:N \definame
4043   \stex_reactivate_macro:N \Definame
4044   \stex_if_smsmode:TF{
4045     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4046   }{
4047     \seq_clear:N \l_tmpa_seq
4048     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4049       \str_if_eq:nnF{ ##1 }{}{
4050         \stex_get_symbol:n { ##1 }
4051         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4052         \l_stex_get_symbol_uri_str
4053     }
4054 }
4055 }
4056 \exp_args:Nnx
4057 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4058     \str_if_empty:NF \sdefinitiontype {
4059         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4060     }
4061     #2
4062     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4063 }
4064 }
4065 \endgroup
4066 \stex_smsmode_do:
4067 }

```

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

33.2 Assertions

sassertion

```

4068
4069 \keys_define:nn {stex / sassertion }{
4070     type      .str_set_x:N = \sassertiontype,
4071     id        .str_set_x:N = \sassertionid,
4072     title     .tl_set:N    = \sassertiontitle ,
4073     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4074     name      .str_set_x:N = \sassertionname
4075 }
4076 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4077     \str_clear:N \sassertiontype
4078     \str_clear:N \sassertionid
4079     \str_clear:N \sassertionname
4080     \clist_clear:N \l__stex_statements_sassertion_for_clist
4081     \tl_clear:N \sassertiontitle
4082     \keys_set:nn { stex / sassertion }{ #1 }
4083 }
4084
4085 %\tl_new:N \g__stex_statements_aftergroup_tl
4086
4087 \NewDocumentEnvironment{sassertion}{0{}}{
4088     \__stex_statements_sassertion_args:n{ #1 }
4089     \stex_if_smsmode:F {
4090         \seq_clear:N \l_tmpa_seq
4091         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4092             \str_if_eq:nnF{ ##1 }{ }{
4093                 \stex_get_symbol:n { ##1 }
4094                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4095                     \l_stex_get_symbol_uri_str
4096                 }
4097             }
4098         }

```

```

4099 \exp_args:Nnnx
4100 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4101 \str_if_empty:NF \sassertiontype {
4102   \stex_annotate_invisible:nnn{type}{\sassertiontype}{\sassertiontype}{}
4103 }
4104 \clist_set:No \l_tmpa_clist \sassertiontype
4105 \tl_clear:N \l_tmpa_tl
4106 \clist_map_inline:Nn \l_tmpa_clist {
4107   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4108     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4109   }
4110 }
4111 \tl_if_empty:NTF \l_tmpa_tl {
4112   \__stex_statements_sassertion_start:
4113 }{
4114   \l_tmpa_tl
4115 }
4116 }
4117 \stex_ref_new_doc_target:n \sassertionid
4118 \stex_smsmode_do:
4119 ){
4120   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4121   \stex_if_smsmode:F {
4122     \clist_set:No \l_tmpa_clist \sassertiontype
4123     \tl_clear:N \l_tmpa_tl
4124     \clist_map_inline:Nn \l_tmpa_clist {
4125       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4126         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4127       }
4128     }
4129     \tl_if_empty:NTF \l_tmpa_tl {
4130       \__stex_statements_sassertion_end:
4131     }{
4132       \l_tmpa_tl
4133     }
4134   }
4135 }
4136 }

```

\stexpatchassertion

```

4137
4138 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4139   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4140     (\sassertiontitle)
4141   }~}
4142 }
4143 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4144
4145 \newcommand\stexpatchassertion[3] [] {
4146   \str_set:Nx \l_tmpa_str{ #1 }
4147   \str_if_empty:NTF \l_tmpa_str {
4148     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4149     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4150   }{

```

```

4151     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4152     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4153   }
4154 }

```

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

`\inlineass` inline:

```

4155 \keys_define:nn {stex / inlineass }{
4156   type      .str_set_x:N = \sassertiontype,
4157   id        .str_set_x:N = \sassertionid,
4158   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4159   name      .str_set_x:N = \sassertionname
4160 }
4161 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4162   \str_clear:N \sassertiontype
4163   \str_clear:N \sassertionid
4164   \str_clear:N \sassertionname
4165   \clist_clear:N \l__stex_statements_sassertion_for_clist
4166   \keys_set:nn { stex / inlineass }{ #1 }
4167 }
4168 \NewDocumentCommand \inlineass { 0{} m } {
4169   \begingroup
4170   \__stex_statements_inlineass_args:n{ #1 }
4171   \stex_ref_new_doc_target:n \sassertionid
4172   \stex_if_smsmode:TF{
4173     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4174   }{
4175     \seq_clear:N \l_tmpa_seq
4176     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4177       \str_if_eq:nnF{ ##1 }{ }{
4178         \stex_get_symbol:n { ##1 }
4179         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4180           \l_stex_get_symbol_uri_str
4181         }
4182       }
4183     }
4184     \exp_args:Nnx
4185     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4186       \str_if_empty:NF \sassertiontype {
4187         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4188       }
4189       #2
4190       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4191     }
4192   }
4193   \endgroup
4194   \stex_smsmode_do:
4195 }

```

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

33.3 Examples

sexample

```

4196 \keys_define:nn {stex / sexample }{
4197   type      .str_set_x:N = \exampletype,
4198   id        .str_set_x:N = \sexampleid,
4199   title     .tl_set:N     = \sexampletitle,
4200   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4201 }
4202 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4203   \str_clear:N \sexampletype
4204   \str_clear:N \sexampleid
4205   \tl_clear:N \sexampletitle
4206   \clist_clear:N \l__stex_statements_sexample_for_clist
4207   \keys_set:nn { stex / sexample }{ #1 }
4208 }
4209
4210
4211 \NewDocumentEnvironment{sexample}{0{}}{
4212   \__stex_statements_sexample_args:n{ #1 }
4213   \stex_if_smsmode:F {
4214     \seq_clear:N \l_tmpa_seq
4215     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4216       \str_if_eq:nnF{ ##1 }{}{
4217         \stex_get_symbol:n { ##1 }
4218         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4219           \l_stex_get_symbol_uri_str
4220         }
4221       }
4222     }
4223     \exp_args:Nnnx
4224     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4225     \str_if_empty:NF \sexampletype {
4226       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4227     }
4228     \clist_set:Nn \l_tmpa_clist \sexampletype
4229     \tl_clear:N \l_tmpa_tl
4230     \clist_map_inline:Nn \l_tmpa_clist {
4231       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4232         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4233       }
4234     }
4235     \tl_if_empty:NTF \l_tmpa_tl {
4236       \__stex_statements_sexample_start:
4237     }{
4238       \l_tmpa_tl
4239     }
4240   }
4241   \stex_ref_new_doc_target:n \sexampleid
4242   \stex_smsmode_do:
4243 }{
4244   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4245   \stex_if_smsmode:F {
4246     \clist_set:Nn \l_tmpa_clist \sexampletype

```

```

4247 \tl_clear:N \l_tmpa_tl
4248 \clist_map_inline:Nn \l_tmpa_clist {
4249   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4250     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4251   }
4252 }
4253 \tl_if_empty:NTF \l_tmpa_tl {
4254   \__stex_statements_sexample_end:
4255 }{
4256   \l_tmpa_tl
4257 }
4258 \end{stex_annotate_env}
4259 }
4260 }

```

`\stexpatchexample`

```

4261
4262 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4263   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplename {
4264     (\sexamplename)
4265   }~}
4266 }
4267 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4268
4269 \newcommand\stexpatchexample[3] [] {
4270   \str_set:Nx \l_tmpa_str{ #1 }
4271   \str_if_empty:NTF \l_tmpa_str {
4272     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4273     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4274   }{
4275     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4276     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4277   }
4278 }

```

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

`\inlineex` inline:

```

4279 \keys_define:nn {stex / inlineex }{
4280   type      .str_set_x:N = \sexamplename,
4281   id        .str_set_x:N = \sexampleid,
4282   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4283   name      .str_set_x:N = \sexamplename
4284 }
4285 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4286   \str_clear:N \sexamplename
4287   \str_clear:N \sexampleid
4288   \str_clear:N \sexamplename
4289   \clist_clear:N \l__stex_statements_sexample_for_clist
4290   \keys_set:nn { stex / inlineex }{ #1 }
4291 }
4292 \NewDocumentCommand \inlineex { 0{} m } {
4293   \begingroup
4294   \__stex_statements_inlineex_args:n{ #1 }

```



```

4295 \stex_ref_new_doc_target:n \sexampleid
4296 \stex_if_smsmode:TF{
4297   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4298 }{
4299   \seq_clear:N \l_tmpa_seq
4300   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4301     \str_if_eq:nnF{ ##1 }{{
4302       \stex_get_symbol:n { ##1 }
4303       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4304         \l_stex_get_symbol_uri_str
4305       }
4306     }
4307   }
4308   \exp_args:Nnx
4309   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}}{
4310     \str_if_empty:NF \sexamplotype {
4311       \stex_annotate_invisible:nnn{type}{\sexamplotype}{
4312     }
4313     #2
4314     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4315   }
4316 }
4317 \endgroup
4318 \stex_smsmode_do:
4319 }

```

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

33.4 Logical Paragraphs

sparagraph

```

4320 \keys_define:nn { stex / sparagraph } {
4321   id      .str_set:N = \sparagraphid ,
4322   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4323   type    .str_set:N  = \sparagraphtype ,
4324   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4325   from    .tl_set:N   = \sparagraphfrom ,
4326   to      .tl_set:N   = \sparagraphto ,
4327   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
4328   name    .str_set:N   = \sparagraphname
4329 }
4330
4331 \cs_new_protected:Nn \stex_sparagraph_args:n {
4332   \tl_clear:N \l_stex_sparagraph_title_tl
4333   \tl_clear:N \sparagraphfrom
4334   \tl_clear:N \sparagraphto
4335   \tl_clear:N \l_stex_sparagraph_start_tl
4336   \str_clear:N \sparagraphid
4337   \str_clear:N \sparagraphtype
4338   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4339   \str_clear:N \sparagraphname
4340   \keys_set:nn { stex / sparagraph }{ #1 }
4341 }

```

```

4342 \newif\if@in@omtext\@in@omtextfalse
4343
4344 \NewDocumentEnvironment {sparagraph} { 0{} } {
4345   \stex_sparagraph_args:n { #1 }
4346   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4347     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4348   }{
4349     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4350   }
4351   \@in@omtexttrue
4352   \stex_if_smsmode:F {
4353     \seq_clear:N \l_tmpa_seq
4354     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4355       \str_if_eq:nnF{ ##1 }{}{
4356         \stex_get_symbol:n { ##1 }
4357         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4358           \l_stex_get_symbol_uri_str
4359         }
4360       }
4361     }
4362     \exp_args:Nnnx
4363     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4364     \str_if_empty:NF \sparagraphtype {
4365       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4366     }
4367     \str_if_empty:NF \sparagraphfrom {
4368       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4369     }
4370     \str_if_empty:NF \sparagraphto {
4371       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4372     }
4373     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4374     \tl_clear:N \l_tmpa_tl
4375     \clist_map_inline:Nn \sparagraphtype {
4376       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4377         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4378       }
4379     }
4380     \tl_if_empty:NTF \l_tmpa_tl {
4381       \__stex_statements_sparagraph_start:
4382     }{
4383       \l_tmpa_tl
4384     }
4385   }
4386   \stex_ref_new_doc_target:n \sparagraphid
4387   \stex_smsmode_do:
4388   \ignorespacesandpars
4389 }{
4390   \stex_if_smsmode:F {
4391     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4392     \tl_clear:N \l_tmpa_tl
4393     \clist_map_inline:Nn \l_tmpa_clist {
4394       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4395         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```

```

4396     }
4397   }
4398   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname} }
4399   \tl_if_empty:NTF \l_tmpa_tl {
4400     \__stex_statements_sparagraph_end:
4401   }{
4402     \l_tmpa_tl
4403   }
4404   \end{stex_annotate_env}
4405 }
4406 }

```

\stexpatchparagraph

```

4407
4408 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4409   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4410     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4411       \titleemph{\l_stex_sparagraph_title_tl}:~
4412     }
4413   }{
4414     \titleemph{\l_stex_sparagraph_start_tl}~
4415   }
4416 }
4417 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4418
4419 \newcommand\stexpatchparagraph[3] [] {
4420   \str_set:Nx \l_tmpa_str{ #1 }
4421   \str_if_empty:NTF \l_tmpa_str {
4422     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4423     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4424   }{
4425     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4426     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4427   }
4428 }
4429
4430 \keys_define:nn { stex / inlinepara } {
4431   id      .str_set_x:N = \sparagraphid ,
4432   type    .str_set_x:N = \sparagraphtype ,
4433   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4434   from    .tl_set:N    = \sparagraphfrom ,
4435   to      .tl_set:N    = \sparagraphto ,
4436   name    .str_set:N   = \sparagraphname
4437 }
4438 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4439   \tl_clear:N \sparagraphfrom
4440   \tl_clear:N \sparagraphto
4441   \str_clear:N \sparagraphid
4442   \str_clear:N \sparagraphtype
4443   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4444   \str_clear:N \sparagraphname
4445   \keys_set:nn { stex / inlinepara }{ #1 }
4446 }
4447 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4448 \begingroup
4449 \__stex_statements_inlinepara_args:n{ #1 }
4450 \stex_ref_new_doc_target:n \sparagraphid
4451 \stex_if_smsmode:TF{
4452   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4453 }{
4454   \seq_clear:N \l_tmpa_seq
4455   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4456     \str_if_eq:nnF{ ##1 }{}{
4457       \stex_get_symbol:n { ##1 }
4458       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4459         \l_stex_get_symbol_uri_str
4460       }
4461     }
4462   }
4463   \exp_args:Nnx
4464   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4465     \str_if_empty:NF \sparagraphtype {
4466       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4467     }
4468     \str_if_empty:NF \sparagraphfrom {
4469       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4470     }
4471     \str_if_empty:NF \sparagraphto {
4472       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4473     }
4474     #2
4475     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4476   }
4477 }
4478 \endgroup
4479 \stex_smsmode_do:
4480 }
4481

```

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

symboldoc

```

4482 \NewDocumentEnvironment{symboldoc}{ m }{
4483   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4484   \seq_clear:N \l_tmpb_seq
4485   \seq_map_inline:Nn \l_tmpa_seq {
4486     \str_if_eq:nnF{ ##1 }{}{
4487       \stex_get_symbol:n { ##1 }
4488       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4489         \l_stex_get_symbol_uri_str
4490       }
4491     }
4492   }
4493   \par
4494   \exp_args:Nnnx
4495   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4496 }{
4497   \end{stex_annotate_env}
4498 }

```

4499 </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

```
4500 <*package>
4501 <@@=stex_sproof>
4502
4503 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4504
```

34.2 Proofs

We first define some keys for the proof environment.

```
4505 \keys_define:nn { stex / spf } {
4506   id          .str_set:N = \l__stex_sproof_spf_id_str,
4507   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4508   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4509   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4510   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4511   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4512   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4513   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4514   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4515   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4516 }
4517 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4518   \str_clear:N \l__stex_sproof_spf_id_str
4519   \tl_clear:N \l__stex_sproof_spf_display_tl
4520   \tl_clear:N \l__stex_sproof_spf_for_tl
4521   \tl_clear:N \l__stex_sproof_spf_from_tl
4522   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4523   \tl_clear:N \l__stex_sproof_spf_type_tl
4524   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4525 \tl_clear:N \l__stex_sproof_spf_continues_tl
4526 \tl_clear:N \l__stex_sproof_spf_functions_tl
4527 \tl_clear:N \l__stex_sproof_spf_method_tl
4528 \keys_set:nn { stex / spf }{ #1 }
4529 }

```

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

```

4530 \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.

```

4531 \newcount\count_ten
4532 \newenvironment{pst@with@label}[1]{
4533   \edef\pst@label{#1}
4534   \advance\count_ten by 1\relax
4535   \count_ten=1
4536 }{
4537   \advance\count_ten by -1\relax
4538 }

```

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

```

4539 \def\the@pst@label{
4540   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4541 }

```

(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.

```

4542 \keys_define:nn { stex / pstlabel }{
4543   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4544   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4545   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4546 }
4547 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4548 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4549 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4550 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4551 }
4552 \__stex_sproof_pstlabel_args:n {}
4553 \newcommand\setpstlabelstyle[1]{
4554   \__stex_sproof_pstlabel_args:n {#1}
4555 }
4556 \newcommand\setpstlabelstyledefault{%
4557   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4558 }

```

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

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

```

4559 \ExplSyntaxOff
4560 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4561 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4562 \def\pst@make@label@short#1#2{#2}
4563 \def\pst@make@label@empty#1#2{}
4564 \ExplSyntaxOn
4565 \def\pstlabelstyle#1{%
4566   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4567 }%
4568 \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.

```

4569 \def\next@pst@label{%
4570   \global\advance\count\count10 by 1%
4571 }%

```

(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

```

4572 \def\sproof@box{
4573   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4574 }
4575 \def\spf@proofend{\sproof@box}
4576 \def\sproofend{
4577   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4578     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4579   }
4580 }
4581 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4582 \def\spf@proofsketch@kw{Proof Sketch}
4583 \def\spf@proof@kw{Proof}
4584 \def\spf@step@kw{Step}

```


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

For the other languages, we set up triggers

```

4585 \AddToHook{begindocument}{
4586   \ltx@ifpackageloaded{babel}{
4587     \makeatletter
4588     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4589     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4590       \input{sproof-ngerman.ldf}
4591     }
4592     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4593       \input{sproof-finnish.ldf}
4594     }
4595     \clist_if_in:NnT \l_tmpa_clist {french}{
4596       \input{sproof-french.ldf}
4597     }
4598     \clist_if_in:NnT \l_tmpa_clist {russian}{
4599       \input{sproof-russian.ldf}
4600     }
4601     \makeatother
4602   }{}
4603 }

```

`spfsketch`

```

4604 \newcommand\spfsketch[2][]{
4605   \__stex_sproof_spf_args:n{#1}
4606   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4607     \titleemph{
4608       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4609         \spf@proofsketch@kw
4610       }{
4611         \l__stex_sproof_spf_type_tl
4612       }
4613     }:
4614   }
4615   {~#2}
4616   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4617   \sproofend
4618 }

```

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

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

```

4619 \newenvironment{spfeq}[2][]{
4620   \__stex_sproof_spf_args:n{#1}
4621   %\sref@target
4622   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4623     \titleemph{
4624       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4625         \spf@proof@kw
4626       }{

```

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

¹⁵EDNOTE: document above

```

4627     \l__stex_sproof_spf_type_tl
4628   }
4629   }:
4630 }
4631 {-#2}
4632 \begin{displaymath}\begin{array}{rcll}
4633 }{
4634 \end{array}\end{displaymath}
4635 }

```

(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.

```

4636 \newenvironment{spf@proof}[2] []{
4637   \l__stex_sproof_spf_args:n{#1}
4638   %\sref@target
4639   \count_ten=10
4640   \par\noindent
4641   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4642     \titleemph{
4643       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4644         \spf@proof@kw
4645       }{
4646         \l__stex_sproof_spf_type_tl
4647       }
4648     }:
4649   }
4650   {-#2}
4651   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4652   \def\pst@label{}
4653   \newcount\pst@count% initialize the labeling mechanism
4654   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4655   }{
4656     \end{pst@with@label}\end{description}
4657   }
4658   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4659   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4660 \newcommand\spfidea[2] []{
4661   \l__stex_sproof_spf_args:n{#1}
4662   \titleemph{
4663     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4664       \l__stex_sproof_spf_type_tl
4665     }:
4666   }-#2
4667   \sproofend
4668 }

```

(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
4669 \newenvironment{spfstep}[1][]{
4670   \_stex_sproof_spf_args:n{#1}
4671   \@in@omtexttrue
4672   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4673     \item[\the@pst@label]
4674   }
4675   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4676     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4677   }
4678   %\sref@label{id{\pst@label}}
4679   \ignorespacesandpars
4680 }{
4681   \next@pst@label\ignorespacesandpars
4682 }

```

sproofcomment

```

4683 \newenvironment{sproofcomment}[1][]{
4684   \_stex_sproof_spf_args:n{#1}
4685   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4686     \item[\the@pst@label]
4687   }
4688 }{
4689   \next@pst@label
4690 }

```

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.

```

4691 \newenvironment{subproof}[2][]{
4692   \_stex_sproof_spf_args:n{#1}
4693   \def\@test{#2}
4694   \ifx\@test\empty\else
4695     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4696       \item[\the@pst@label]
4697     }{#2}
4698   \fi
4699   \begin{pst@with@label}{\pst@label,\number\count_ten}
4700 }{
4701   \end{pst@with@label}\next@pst@label
4702 }

```

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

```

4703 \newenvironment{spfcases}[2][]{
4704   \def\@test{#1}
4705   \ifx\@test\empty
4706     \begin{subproof}[method=by-cases]{#2}

```

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

```

4707 \else
4708   \begin{subproof}[#1,method=by-cases]{#2}
4709 \fi
4710 }{
4711   \end{subproof}
4712 }

```

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

```

4713 \newenvironment{spfcase}[2] [] {
4714   \__stex_sproof_spf_args:n{#1}
4715   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4716     \item[\the@pst@label]
4717   }
4718   \def\@test{#2}
4719   \ifx\@test\@empty
4720   \else
4721     {\titleemph{#2}:~}
4722   \fi
4723   \begin{pst@with@label}{\pst@label,\number\count_ten}
4724 }{
4725   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4726     \sproofend
4727   }
4728   \end{pst@with@label}
4729   \next@pst@label
4730 }

```

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

```

4731 \newcommand\spfcasesketch[3] [] {
4732   \__stex_sproof_spf_args:n{#1}
4733   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4734     \item[\the@pst@label]
4735   }
4736   \def\@test{#2}
4737   \ifx\@test\@empty
4738   \else
4739     {\titleemph{#2}:~}
4740   \fi#3
4741   \next@pst@label
4742 }%

```

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.

```

4743 \keys_define:nn { stex / just }{
4744   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4745   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4746   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4747   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4748 }

```

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

`justification`

4749 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4750 `\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.

4751 `\newcommand\justarg[2] [] {#2}`

4752 `\</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

```
4753 <*package>
4754
4755 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4756
4757 <@@=stex_others>
      Warnings and error messages
4758 % None

\MSC Math subject classifier

4759 \NewDocumentCommand \MSC {m} {
4760   % TODO
4761 }

(End definition for \MSC. This function is documented on page 21.)
      Patching tikzinput, if loaded
4762 \@ifpackageloaded{tikzinput}{
4763   \RequirePackage{stex-tikzinput}
4764 }{}
4765 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4766 <*package>
4767 <@@=stex_modules>
4768
4769 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4770
4771 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4772 \begingroup
4773 \stex_module_setup:nn{
4774   ns=\c_stex_metatheory_ns_str,
4775   meta=NONE
4776 }{Metatheory}
4777 \stex_reactivate_macro:N \symdecl
4778 \stex_reactivate_macro:N \notation
4779 \stex_reactivate_macro:N \symdef
4780 \ExplSyntaxOff
4781 \csname stex_suppress_html:n\endcsname{
4782   % is-a (a:A, a \in A, a is an A, etc.)
4783   \symdecl[args=ai]{isa}
4784   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4785   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4786   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4787
4788   % bind (\forall, \Pi, \lambda etc.)
4789   \symdecl[args=Bi]{bind}
4790   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4791   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4792   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4793
4794   % dummy variable
4795   \symdecl{dummyvar}
4796   \notation[underscore]{dummyvar}{\comp\_}
4797   \notation[dot]{dummyvar}{\comp\cdot}
4798   \notation[dash]{dummyvar}{\comp{\rm --}}
4799
4800   %fromto (function space, Hom-set, implication etc.)
```

```

4801 \symdecl[args=ai]{fromto}
4802 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4803 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4804
4805 % mapto (lambda etc.)
4806 \%symdecl[args=Bi]{mapto}
4807 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4808 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4809 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4810
4811 % function/operator application
4812 \symdecl[args=ia]{apply}
4813 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4814 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4815
4816 % ‘type’ of all collections (sets, classes, types, kinds)
4817 \symdecl{collection}
4818 \notation[U]{collection}{\comp{\mathcal{U}}}
4819 \notation[set]{collection}{\comp{\textsf{Set}}}
4820
4821 % sequences
4822 \symdecl[args=1]{seqtype}
4823 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4824
4825 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4826 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4827
4828 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4829 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4830 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4831
4832 % letin (‘let’, local definitions, variable substitution)
4833 \symdecl[args=bii]{letin}
4834 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4835 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4836 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4837
4838 % structures
4839 \symdecl*[args=1]{module-type}
4840 \notation{module-type}{\mathtt{MOD} #1}
4841 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4842 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4843
4844 }
4845 \ExplSyntaxOn
4846 \stex_add_to_current_module:n{
4847   \let\nappa\apply
4848   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4849   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4850   \def\livar{\csname sequence-index\endcsname[li]}
4851   \def\uivar{\csname sequence-index\endcsname[ui]}
4852   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4853   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4854   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```



```
4855 }  
4856 \__stex_modules_end_module:  
4857 \endgroup  
4858 \</package>
```

Chapter 37

Tikzinput Implementation

```
4859 <*package>
4860
4861 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4862
4863 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4864 \RequirePackage{l3keys2e}
4865
4866 \keys_define:nn { tikzinput } {
4867   image .bool_set:N = \c_tikzinput_image_bool,
4868   image .default:n = false ,
4869   unknown .code:n = {}
4870 }
4871
4872 \ProcessKeysOptions { tikzinput }
4873
4874 \bool_if:NTF \c_tikzinput_image_bool {
4875   \RequirePackage{graphicx}
4876
4877   \providecommand\usetikzlibrary[]{}
4878   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4879 }{
4880   \RequirePackage{tikz}
4881   \RequirePackage{standalone}
4882
4883   \newcommand \tikzinput [2] [] {
4884     \setkeys{Gin}{#1}
4885     \ifx \Gin@ewidth \Gin@exclamation
4886       \ifx \Gin@eheight \Gin@exclamation
4887         \input { #2 }
4888       \else
4889         \resizebox{!}{ \Gin@eheight }{
4890           \input { #2 }
4891         }
4892       \fi
4893     \else
4894       \ifx \Gin@eheight \Gin@exclamation
4895         \resizebox{ \Gin@ewidth }{!}{
4896           \input { #2 }
```

```

4897     }
4898     \else
4899         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4900             \input { #2 }
4901         }
4902     \fi
4903 \fi
4904 }
4905 }
4906
4907 \newcommand \ctikzinput [2] [] {
4908     \begin{center}
4909         \tikzinput [ #1 ] { #2 }
4910     \end{center}
4911 }
4912
4913 \@ifpackageloaded{stex}{
4914     \RequirePackage{stex-tikzinput}
4915 }{}
4916
4917 </package>
4918 <*stex>
4919 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4920 \RequirePackage{stex}
4921 \RequirePackage{tikzinput}
4922
4923 \newcommand\mhtikzinput [2] [] {%
4924     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4925     \stex_in_repository:nn\Gin@mhrepos{
4926         \tikzinput [ #1 ] {\mhp{##1}{#2}}
4927     }
4928 }
4929 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
4930 </stex>

```

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

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.

```
4931 \*cls)
4932 \@@=document_structure)
4933 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4934 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

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`

```
4935 \keys_define:nn{ document-structure / pkg }{
4936   class      .str_set_x:N = \c_document_structure_class_str,
4937   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4938   report     .code:n      = {
4939     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4940     \str_set:Nn \c_document_structure_class_str {report}
4941   },
4942   book       .code:n      = {
4943     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4944     \str_set:Nn \c_document_structure_class_str {book}
4945   },
4946   bookpart   .code:n      = {
4947     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4948     \str_set:Nn \c_document_structure_class_str {book}
4949     \str_set:Nn \c_document_structure_topsect_str {chapter}
4950   },
```

```

4951 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4952 unknown     .code:n      = {
4953   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4954 }
4955 }
4956 \ProcessKeysOptions{ document-structure / pkg }
4957 \str_if_empty:NT \c_document_structure_class_str {
4958   \str_set:Nn \c_document_structure_class_str {article}
4959 }
4960 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4961   {\c_document_structure_class_str}
4962

```

38.3 Beefing up the document environment

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

```

4963 \RequirePackage{document-structure}
4964 \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_EXML.¹⁸

```

4965 \keys_define:nn { document-structure / document }{
4966   id .str_set_x:N = \c_document_structure_document_id_str
4967 }
4968 \let\__document_structure_orig_document=\document
4969 \renewcommand{\document}[1][]{
4970   \keys_set:nn{ document-structure / document }{ #1 }
4971   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4972   \__document_structure_orig_document
4973 }

```

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

```

4974 }
4975 \</cls>

```

38.4 Implementation: document-structure Package

```

4976 \<*package>
4977 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
4978 \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

```

4979
4980 \keys_define:nn{ document-structure / pkg }{
4981   class      .str_set_x:N = \c_document_structure_class_str,
4982   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4983   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4984 }
4985 \ProcessKeysOptions{ document-structure / pkg }
4986 \str_if_empty:NT \c_document_structure_class_str {
4987   \str_set:Nn \c_document_structure_class_str {article}
4988 }
4989 \str_if_empty:NT \c_document_structure_topsect_str {
4990   \str_set:Nn \c_document_structure_topsect_str {section}
4991 }

```

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

```

4992 \RequirePackage{xspace}
4993 \RequirePackage{comment}
4994 \AddToHook{begindocument}{
4995   \ltx@ifpackageloaded{babel}{
4996     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4997     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4998       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4999     }
5000   }{}
5001 }

```

`\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.

```

5002 \int_new:N \l_document_structure_section_level_int
5003 \str_case:VnF \c_document_structure_topsect_str {
5004   {part}}{
5005     \int_set:Nn \l_document_structure_section_level_int {0}
5006   }
5007   {chapter}{
5008     \int_set:Nn \l_document_structure_section_level_int {1}
5009   }
5010 }{
5011   \str_case:VnF \c_document_structure_class_str {
5012     {book}}{
5013       \int_set:Nn \l_document_structure_section_level_int {0}
5014     }
5015     {report}}{
5016       \int_set:Nn \l_document_structure_section_level_int {0}
5017     }
5018   }{
5019     \int_set:Nn \l_document_structure_section_level_int {2}
5020   }
5021 }

```

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

```
5022 \def\current@section@level{document}%
5023 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5024 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5025 \cs_new_protected:Npn \skipomgroup {
5026   \ifcase\l_document_structure_section_level_int
5027   \or\stepcounter{part}
5028   \or\stepcounter{chapter}
5029   \or\stepcounter{section}
5030   \or\stepcounter{subsection}
5031   \or\stepcounter{subsubsection}
5032   \or\stepcounter{paragraph}
5033   \or\stepcounter{subparagraph}
5034   \fi
5035 }
```

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

`blindomgroup`

```
5036 \newcommand\at@begin@blindomgroup[1]{%
5037 \newenvironment{blindomgroup}
5038 {
5039   \int_incr:N\l_document_structure_section_level_int
5040   \at@begin@blindomgroup\l_document_structure_section_level_int
5041 }{}}
```

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

```
5042 \newcommand\omgroup@nonum[2]{
5043   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5044   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5045 }
```

(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.

```
5046 \newcommand\omgroup@num[2]{
```

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

```

5047 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5048   \@nameuse{#1}{#2}
5049 }{
5050   \cs_if_exist:NTF\rdfmata@sectioning{
5051     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5052   }{
5053     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5054   }
5055 }
5056 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5057 }

```

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

omgroup

```

5058 \keys_define:nn { document-structure / omgroup }{
5059   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5060   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5061   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
5062   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5063   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
5064   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
5065   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
5066   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
5067   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
5068   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5069 }
5070 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
5071   \str_clear:N \l__document_structure_omgroup_id_str
5072   \str_clear:N \l__document_structure_omgroup_date_str
5073   \clist_clear:N \l__document_structure_omgroup_creators_clist
5074   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5075   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5076   \tl_clear:N \l__document_structure_omgroup_type_tl
5077   \tl_clear:N \l__document_structure_omgroup_short_tl
5078   \tl_clear:N \l__document_structure_omgroup_display_tl
5079   \tl_clear:N \l__document_structure_omgroup_intro_tl
5080   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5081   \keys_set:nn { document-structure / omgroup } { #1 }
5082 }

```

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

```

5083 \newif\if@mainmatter\@mainmattertrue
5084 \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.

```

5085 \keys_define:nn { document-structure / sectioning }{
5086   name .str_set_x:N = \l__document_structure_sect_name_str ,
5087   ref   .str_set_x:N = \l__document_structure_sect_ref_str ,
5088   clear .bool_set:N  = \l__document_structure_sect_clear_bool ,
5089   num   .bool_set:N  = \l__document_structure_sect_num_bool ,
5090 }

```



```

5091 \cs_new_protected:Nn \__document_structure_sect_args:n {
5092   \str_clear:N \l__document_structure_sect_name_str
5093   \str_clear:N \l__document_structure_sect_ref_str
5094   \bool_set_false:N \l__document_structure_sect_clear_bool
5095   \bool_set_false:N \l__document_structure_sect_num_bool
5096   \keys_set:nn { document-structure / sectioning } { #1 }
5097 }
5098 \newcommand\omdoc@sectioning[3][]{
5099   \__document_structure_sect_args:n {#1}
5100   \let\omdoc@sect@name\l__document_structure_sect_name_str
5101   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5102   \if@mainmatter% numbering not overridden by frontmatter, etc.
5103     \bool_if:NTF \l__document_structure_sect_num_bool {
5104       \omgroup@num{#2}{#3}
5105     }{
5106       \omgroup@nonum{#2}{#3}
5107     }
5108     \def\current@section@level{\omdoc@sect@name}
5109   \else
5110     \omgroup@nonum{#2}{#3}
5111   \fi
5112 }% 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.

```

5113 \newcommand\omgroup@redefine@addtocontents[1]{%
5114   %\edef\__document_structureimport{#1}%
5115   %\@for\@I:=\__document_structureimport\do{%
5116     \edef\@path{\csname module@\@I @path\endcsname}%
5117     %\@ifundefined{tf@toc}\relax%
5118     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5119   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5120   %\def\addcontentsline##1##2##3{%
5121     \addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5122   %\else% hyperref.sty not loaded
5123   %\def\addcontentsline##1##2##3{%
5124     \addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5125   %\fi
5126 }% 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.

```

5127 \int_new:N \l_document_structure_omgroup_level_int
5128 \newenvironment{omgroup}[2][]{% keys, title
5129 {
5130   \__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.

```

5131 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5132   \omgroup@redefine@addtocontents{
5133     %\@ifundefined{module@id}\used@modules%
5134     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5135     }
5136 }

```

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

```

5137 \int_incr:N \l_document_structure_omgroup_level_int
5138 \int_incr:N \l_document_structure_section_level_int
5139 \ifcase\l_document_structure_section_level_int
5140   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5141   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5142   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5143   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5144   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5145   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5146   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5147 \fi
5148 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5149 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5150 }% for customization
5151 {}

```

and finally, we localize the sections

```

5152 \newcommand\omdoc@part@kw{Part}
5153 \newcommand\omdoc@chapter@kw{Chapter}
5154 \newcommand\omdoc@section@kw{Section}
5155 \newcommand\omdoc@subsection@kw{Subsection}
5156 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5157 \newcommand\omdoc@paragraph@kw{paragraph}
5158 \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`

```

5159 \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).

```

5160 \cs_if_exist:NTF\frontmatter{
5161   \let\__document_structure_orig_frontmatter\frontmatter
5162   \let\frontmatter\relax
5163 }{
5164   \tl_set:Nn\__document_structure_orig_frontmatter{
5165     \clearpage
5166     \@mainmatterfalse
5167     \pagenumbering{roman}
5168   }
5169 }

```

```

5170 \cs_if_exist:NTF\backmatter{
5171   \let\__document_structure_orig_backmatter\backmatter
5172   \let\backmatter\relax
5173 }{
5174   \tl_set:Nn\__document_structure_orig_backmatter{
5175     \clearpage
5176     \@mainmatterfalse
5177     \pagenumbering{roman}
5178   }
5179 }

```

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.

```

5180 \newenvironment{frontmatter}{
5181   \__document_structure_orig_frontmatter
5182 }{
5183   \cs_if_exist:NTF\mainmatter{
5184     \mainmatter
5185   }{
5186     \clearpage
5187     \@mainmattertrue
5188     \pagenumbering{arabic}
5189   }
5190 }

```

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

```

5191 \newenvironment{backmatter}{
5192   \__document_structure_orig_backmatter
5193 }{
5194   \cs_if_exist:NTF\mainmatter{
5195     \mainmatter
5196   }{
5197     \clearpage
5198     \@mainmattertrue
5199     \pagenumbering{arabic}
5200   }
5201 }

```

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

```

5202 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5203 \def \c__document_structure_document_str{document}
5204 \newcommand\afterprematurestop{}
5205 \def\prematurestop@endomgroup{
5206   \unless\ifx\@currenvir\c__document_structure_document_str
5207     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
5208     \expandafter\prematurestop@endomgroup
5209   \fi
5210 }

```

```

5211 \providecommand\prematurestop{
5212   \message{Stopping~sTeX~processing~prematurely}
5213   \prematurestop@endomgroup
5214   \afterprematurestop
5215   \end{document}
5216 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5217 \RequirePackage{etoolbox}
5218 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5219 \newrobustcmd\useSGvar[1]{%
5220   \@ifundefined{sTeX@Gvar@#1}
5221   {\PackageError{document-structure}
5222    {The sTeX Global variable #1 is undefined}
5223    {set it with \protect\setSGvar}}
5224   \@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.

```

5225 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5226   \@ifundefined{sTeX@Gvar@#1}
5227   {\PackageError{document-structure}
5228    {The sTeX Global variable #1 is undefined}
5229    {set it with \protect\setSGvar}}
5230   {\expandafter\ifx\cename 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.

```
5231 \*cls)
5232 \@@=notesslides)
5233 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5234 \RequirePackage{l3keys2e,expl-keystr-compatible}
5235
5236 \keys_define:nn{notesslides / cls}{
5237   class .code:n = {
5238     \PassOptionsToClass{\CurrentOption}{omdoc}
5239     \str_if_eq:nnT{#1}{book}{
5240       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5241     }
5242     \str_if_eq:nnT{#1}{report}{
5243       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5244     }
5245   },
5246   notes .bool_set:N = \c__notesslides_notes_bool ,
5247   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5248   unknown .code:n = {
5249     \PassOptionsToClass{\CurrentOption}{omdoc}
5250     \PassOptionsToClass{\CurrentOption}{beamer}
5251     \PassOptionsToPackage{\CurrentOption}{notesslides}
5252   }
5253 }
5254 \ProcessKeysOptions{ notesslides / cls }
5255 \bool_if:NTF \c__notesslides_notes_bool {
5256   \PassOptionsToPackage{notes=true}{notesslides}
5257 }{
5258   \PassOptionsToPackage{notes=false}{notesslides}
5259 }
5260 \</cls)
```

now we do the same for the notesslides package.

```

5261 <*package>
5262 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5263 \RequirePackage{13keys2e,expl-keystr-compat}
5264
5265 \keys_define:nn{notesslides / pkg}{
5266   topsect          .str_set_x:N = \c__notesslides_topsect_str,
5267   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
5268   notes            .bool_set:N = \c__notesslides_notes_bool ,
5269   slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
5270   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
5271   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
5272   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
5273   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
5274   unknown          .code:n       = {
5275     \PassOptionsToClass{\CurrentOption}{stex}
5276     \PassOptionsToClass{\CurrentOption}{tikzinput}
5277   }
5278 }
5279 \ProcessKeysOptions{ notesslides / pkg }
5280 \newif\ifnotes
5281 \bool_if:NTF \c__notesslides_notes_bool {
5282   \notesttrue
5283 }{
5284   \notesfalse
5285 }
5286

```

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

```

5287 \str_if_empty:NTF \c__notesslides_topsect_str {
5288   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
5289 }{
5290   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
5291 }
5292 </package>

```

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

```

5293 <*cls>
5294 \bool_if:NTF \c__notesslides_notes_bool {
5295   \LoadClass{document-structure}
5296 }{
5297   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5298   \newcounter{Item}
5299   \newcounter{paragraph}
5300   \newcounter{subparagraph}
5301   \newcounter{Hfootnote}
5302   \RequirePackage{document-structure}
5303 }

```

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

```

5304 \RequirePackage{notesslides}
5305 </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.

```

5306 \*package>
5307 \bool_if:NT \c__notesslides_notes_bool {
5308   \RequirePackage{a4wide}
5309   \RequirePackage{marginnote}
5310   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5311   \RequirePackage{mdframed}
5312   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5313   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5314 }
5315 \RequirePackage{stex-tikzinput}
5316 \RequirePackage{etoolbox}
5317 \RequirePackage{amssymb}
5318 \RequirePackage{amsmath}
5319 \RequirePackage{comment}
5320 \RequirePackage{textcomp}
5321 \RequirePackage{url}
5322 \RequirePackage{graphicx}
5323 \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

```

5324 \bool_if:NT \c__notesslides_notes_bool {
5325   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5326 }

```

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

```

5327 \newcounter{slide}
5328 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5329 \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.

```

5330 \bool_if:NTF \c__notesslides_notes_bool {
5331   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5332 }{
5333   \excludecomment{note}
5334 }

```

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.

```
5335 \bool_if:NT \c__notesslides_notes_bool {
5336   \newlength{\slideframewidth}
5337   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5338 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5339   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5340     \bool_set_true:N #1
5341   }{
5342     \bool_set_false:N #1
5343   }
5344 }
5345 \keys_define:nn{notesslides / frame}{
5346   label .str_set_x:N = \l__notesslides_frame_label_str,
5347   allowframebreaks .code:n = {
5348     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5349   },
5350   allowdisplaybreaks .code:n = {
5351     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5352   },
5353   fragile .code:n = {
5354     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5355   },
5356   shrink .code:n = {
5357     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5358   },
5359   squeeze .code:n = {
5360     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5361   },
5362   t .code:n = {
5363     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5364   },
5365 }
5366 \cs_new_protected:Nn \__notesslides_frame_args:n {
5367   \str_clear:N \l__notesslides_frame_label_str
5368   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5369   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5370   \bool_set_true:N \l__notesslides_frame_fragile_bool
5371   \bool_set_true:N \l__notesslides_frame_shrink_bool
5372   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5373   \bool_set_true:N \l__notesslides_frame_t_bool
5374   \keys_set:nn { notesslides / frame }{ #1 }
5375 }
```

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

```
5376 \renewenvironment{frame}[1][]{
5377   \__notesslides_frame_args:n{#1}
5378   \sffamily
5379   \stepcounter{slide}
5380   \def\@currentlabel{\theslide}
5381   \str_if_empty:NF \l__notesslides_frame_label_str {
5382     \label{\l__notesslides_frame_label_str}
```


5383 }

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

5384 \def\itemize@level{outer}
5385 \def\itemize@outer{outer}
5386 \def\itemize@inner{inner}
5387 \renewcommand\newpage{\addtocounter{framenum}{1}}
5388 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5389 \renewenvironment{itemize}{
5390 \ifx\itemize@level\itemize@outer
5391 \def\itemize@label{\\$ \rhd\$}
5392 \fi
5393 \ifx\itemize@level\itemize@inner
5394 \def\itemize@label{\\$ \scriptstyle \rhd\$}
5395 \fi
5396 \begin{list}
5397 {\itemize@label}
5398 {\setlength{\labelsep}{.3em}
5399 \setlength{\labelwidth}{.5em}
5400 \setlength{\leftmargin}{1.5em}
5401 }
5402 \edef\itemize@level{\itemize@inner}
5403 }{
5404 \end{list}
5405 }

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

5406 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5407 }{
5408 \medskip\miko@slidelabel\end{mdframed}
5409 }

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

\frametitle

5410 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5411 }

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

EdN:21

\pause 21

5412 \bool_if:NT \c__notesslides_notes_bool {
5413 \newcommand\pause{
5414 }

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

nparagraph

5415 \bool_if:NTF \c__notesslides_notes_bool {
5416 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5417 }{
5418 \excludecomment{nparagraph}
5419 }

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

```

nomgroup
5420 \bool_if:NTF \c__notesslides_notes_bool {
5421   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5422 }{
5423   \excludecomment{nomgroup}
5424 }

ndefinition
5425 \bool_if:NTF \c__notesslides_notes_bool {
5426   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5427 }{
5428   \excludecomment{ndefinition}
5429 }

nassertion
5430 \bool_if:NTF \c__notesslides_notes_bool {
5431   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5432 }{
5433   \excludecomment{nassertion}
5434 }

nsproof
5435 \bool_if:NTF \c__notesslides_notes_bool {
5436   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5437 }{
5438   \excludecomment{nproof}
5439 }

nexample
5440 \bool_if:NTF \c__notesslides_notes_bool {
5441   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5442 }{
5443   \excludecomment{nexample}
5444 }

\inputref@*skip We customize the hooks for in \inputref.
5445 \def\inputref@preskip{\smallskip}
5446 \def\inputref@postskip{\medskip}

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

\inputref*
5447 \let\orig@inputref\inputref
5448 \def\inputref{\@ifstar\ninputref\orig@inputref}
5449 \newcommand\ninputref[2] [] {
5450   \bool_if:NT \c__notesslides_notes_bool {
5451     \orig@inputref[#1]{#2}
5452   }
5453 }

(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>}`.

```

5454 \newlength{\slidelogoheight}
5455
5456 \bool_if:NTF \c__notesslides_notes_bool {
5457   \setlength{\slidelogoheight}{.4cm}
5458 }{
5459   \setlength{\slidelogoheight}{1cm}
5460 }
5461 \newsavebox{\slidelogo}
5462 \sbox{\slidelogo}{\TeX}
5463 \newrobustcmd{\setslidelogo}[1]{
5464   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5465 }
```

(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.

```

5466 \def\source{Michael Kohlhase}% customize locally
5467 \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.

```

5468 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5469 \newsavebox{\cclogo}
5470 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5471 \newif\ifcchref\cchreffalse
5472 \AtBeginDocument{
5473   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5474 }
5475 \def\licensing{
5476   \ifcchref
5477     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5478   \else
5479     {\usebox{\cclogo}}
5480   \fi
5481 }
5482 \newrobustcmd{\setlicensing}[2][]{
5483   \def@url{#1}
5484   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5485   \ifx@url@empty
5486     \def\licensing{{\usebox{\cclogo}}}
5487   \else
5488     \def\licensing{
```

```

5489     \ifcchref
5490     \href{#1}{\usebox{\cclogo}}
5491     \else
5492     {\usebox{\cclogo}}
5493     \fi
5494   }
5495 \fi
5496 }

```

(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

```

5497 \newrobustcmd\miko@slidelabel{
5498   \vbox to \slidelogoheight{
5499     \vss\hbox to \slidewidth
5500     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5501   }
5502 }

```

(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.

```

5503 \def\Gin@mhrepos{}
5504 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5505 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5506 \newrobustcmd\frameimage[2][]{
5507   \stepcounter{slide}
5508   \bool_if:NT \c__notesslides_frameimages_bool {
5509     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5510     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5511     \begin{center}
5512       \bool_if:NTF \c__notesslides_fiboxed_bool {
5513         \fbox{
5514           \ifx\Gin@ewidth\@empty
5515             \ifx\Gin@mhrepos\@empty
5516               \mhgraphics[width=\slidewidth,#1]{#2}
5517             \else
5518               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5519             \fi
5520           \else% Gin@ewidth empty
5521             \ifx\Gin@mhrepos\@empty
5522               \mhgraphics[#1]{#2}
5523             \else
5524               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5525             \fi
5526           \fi% Gin@ewidth empty
5527         }
5528       }{
5529         \ifx\Gin@ewidth\@empty

```

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

```

5530         \ifx\Gin@mhrepos\empty
5531             \mhgraphics[width=\slidewidth,#1]{#2}
5532         \else
5533             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5534         \fi
5535         \ifx\Gin@mhrepos\empty
5536             \mhgraphics[#1]{#2}
5537         \else
5538             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5539         \fi
5540     \fi% Gin@ewidth empty
5541 }
5542 \end{center}
5543 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5544 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5545 }
5546 } % 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.

```

5547 \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

```

5548 \AddToHook{begindocument}{
5549     \definecolor{green}{rgb}{0,.5,0}
5550     \definecolor{purple}{cmyk}{.3,1,0,.17}
5551 }

```

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`.

```

5552 % \def\STpresent#1{\textcolor{blue}{#1}}
5553 \def\defemph#1{\textcolor{magenta}{#1}}
5554 \def\symrefemph#1{\textcolor{cyan}{#1}}
5555 \def\compemph#1{\textcolor{blue}{#1}}
5556 \def\titleemph#1{\textcolor{blue}{#1}}
5557 \def\__omtext_lec#1(\textcolor{green}{#1})

```

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

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

```

5558 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5559 \def\smalltextwarning{
5560     \pgfuseimage{miko@small@dbend}
5561     \xspace
5562 }
5563 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5564 \newrobustcmd\textwarning{
5565   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5566   \xspace
5567 }
5568 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5569 \newrobustcmd\bigtextwarning{
5570   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5571   \xspace
5572 }

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

5573 \newrobustcmd\putgraphicsat[3]{
5574   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5575 }
5576 \newrobustcmd\putat[2]{
5577   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5578 }

```

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`.

```

5579 \bool_if:NT \c__notesslides_sectocframes_bool {
5580   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5581     \newcounter{chapter}\counterwithin*{section}{chapter}
5582   }{
5583     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5584       \newcounter{chapter}\counterwithin*{section}{chapter}
5585     }
5586   }
5587 }

```

`\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

5588 \def\part@prefix{}
5589 \@ifpackageloaded{document-structure}{}{
5590   \str_case:VnF \__notesslidesstopsect {
5591     {part}{
5592       \int_set:Nn \l_document_structure_section_level_int {0}
5593       \def\thesection{\arabic{chapter}.\arabic{section}}
5594       \def\part@prefix{\arabic{chapter}.}
5595     }
5596     {chapter}{
5597       \int_set:Nn \l_document_structure_section_level_int {1}
5598       \def\thesection{\arabic{chapter}.\arabic{section}}
5599       \def\part@prefix{\arabic{chapter}.}
5600     }
5601   }{
5602     \int_set:Nn \l_document_structure_section_level_int {2}
5603     \def\part@prefix{}

```

```

5604 }
5605 }
5606
5607 \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

```

5608 \renewenvironment{omgroup}[2][]{
5609   \__document_structure_omgroup_args:n { #1 }
5610   \int_incr:N \l_document_structure_omgroup_level_int
5611   \int_incr:N \l_document_structure_section_level_int
5612   \bool_if:NT \c__notesslides_sectocframes_bool {
5613     \stepcounter{slide}
5614     \begin{frame}[noframenumbering]
5615     \vfill\Large\centering
5616     \red{
5617       \ifcase\l_document_structure_section_level_int\or
5618         \stepcounter{part}
5619         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5620         \def\currentsectionlevel{\omdoc@part@kw}
5621       \or
5622         \stepcounter{chapter}
5623         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5624         \def\currentsectionlevel{\omdoc@chapter@kw}
5625       \or
5626         \stepcounter{section}
5627         \def\__notesslideslabel{\part@prefix\arabic{section}}
5628         \def\currentsectionlevel{\omdoc@section@kw}
5629       \or
5630         \stepcounter{subsection}
5631         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5632         \def\currentsectionlevel{\omdoc@subsection@kw}
5633       \or
5634         \stepcounter{subsubsection}
5635         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5636         \def\currentsectionlevel{\omdoc@subsubsection@kw}
5637       \or
5638         \stepcounter{paragraph}
5639         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5640         \def\currentsectionlevel{\omdoc@paragraph@kw}
5641       \else
5642         \def\__notesslideslabel{}
5643         \def\currentsectionlevel{\omdoc@paragraph@kw}
5644       \fi% end ifcase
5645       \__notesslideslabel%\sref@label@id\__notesslideslabel
5646       \quad #2%
5647     }%
5648     \vfill%
5649     \end{frame}%
5650   }
5651   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5652 }{}
5653 }

```

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

```

5654 \def\inserttheorembodyfont{\normalfont}
5655 %\bool_if:NF \c__notesslides_notes_bool {
5656 % \defbeamertemplate{theorem begin}{miko}
5657 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5658 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5659 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5660 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5661 % \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.

```

5662 % \expandafter\def\csname Parent2\endcsname{}
5663 %}
5664
5665 \AddToHook{begindocument}{% this does not work for some reasons
5666 \setbeamertemplate{theorems}[ams style]
5667 }
5668 \bool_if:NT \c__notesslides_notes_bool {
5669 \renewenvironment{columns}[1][{}%
5670 \par\noindent%
5671 \begin{minipage}%
5672 \slidewidth\centering\leavevmode%
5673 }{}%
5674 \end{minipage}\par\noindent%
5675 }%
5676 \newsavebox\columnbox%
5677 \renewenvironment<>{column}[2][{}%
5678 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5679 }{}%
5680 \end{minipage}\end{lrbox}\usebox\columnbox%
5681 }%
5682 }
5683 \bool_if:NTF \c__notesslides_noproblems_bool {
5684 \newenvironment{problems}{}{}
5685 }{
5686 \excludecomment{problems}
5687 }

```

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.

```

5688 \gdef\printexcursions{}
5689 \newcommand\excursionref[2]{% label, text
5690 \bool_if:NT \c__notesslides_notes_bool {

```



```

5691 \begin{sparagraph}[title=Excursion]
5692 #2 \sref[fallback=the appendix]{#1}.
5693 \end{sparagraph}
5694 }
5695 }
5696 \newcommand\activate@excursion[2][] {
5697 \gappto\printexcursions{\inputref{#1}{#2}}
5698 }
5699 \newcommand\excursion[4][] {% repos, label, path, text
5700 \bool_if:NT \c__notesslides_notes_bool {
5701 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
5702 }
5703 }

```

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

\excursiongroup

```

5704 \keys_define:nn{notesslides / excursiongroup }{
5705 id .str_set_x:N = \l__notesslides_excursion_id_str,
5706 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5707 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5708 }
5709 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5710 \tl_clear:N \l__notesslides_excursion_intro_tl
5711 \str_clear:N \l__notesslides_excursion_id_str
5712 \str_clear:N \l__notesslides_excursion_mhrepos_str
5713 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5714 }
5715 \newcommand\excursiongroup[1][] {
5716 \__notesslides_excursion_args:n{ #1 }
5717 \ifdefempty\printexcursions{}% only if there are excursions
5718 {\begin{note}
5719 \begin{omgroup}[#1]{Excursions}%
5720 \ifdefempty\l__notesslides_excursion_intro_tl}{
5721 \inputref[\l__notesslides_excursion_mhrepos_str]{
5722 \l__notesslides_excursion_intro_tl
5723 }
5724 }
5725 \printexcursions%
5726 \end{omgroup}
5727 \end{note}}
5728 }
5729 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5730 \end{package}

```

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

Chapter 40

The Implementation

40.1 Package Options

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

```
5731 \*package>
5732 \@@=problems>
5733 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5734 \RequirePackage{l3keys2e,expl-keystr-compat}
5735
5736 \keys_define:nn { problem / pkg }{
5737   notes      .default:n    = { true },
5738   notes      .bool_set:N   = \c__problems_notes_bool,
5739   gnotes     .default:n    = { true },
5740   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5741   hints      .default:n    = { true },
5742   hints      .bool_set:N   = \c__problems_hints_bool,
5743   solutions  .default:n    = { true },
5744   solutions  .bool_set:N   = \c__problems_solutions_bool,
5745   pts        .default:n    = { true },
5746   pts        .bool_set:N   = \c__problems_pts_bool,
5747   min        .default:n    = { true },
5748   min        .bool_set:N   = \c__problems_min_bool,
5749   boxed      .default:n    = { true },
5750   boxed      .bool_set:N   = \c__problems_boxed_bool,
5751   unknown    .code:n       = {}
5752 }
5753 \newif\ifsolutions
5754
5755 \ProcessKeysOptions{ problem / pkg }
5756 \bool_if:NTF \c__problems_solutions_bool {
5757   \solutionstrue
5758 }{
5759   \solutionsfalse
5760 }
```

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

```
5761 \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.

```
5762 \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.

```
5763 \def\prob@problem@kw{Problem}
5764 \def\prob@solution@kw{Solution}
5765 \def\prob@hint@kw{Hint}
5766 \def\prob@note@kw{Note}
5767 \def\prob@gnote@kw{Grading}
5768 \def\prob@pt@kw{pt}
5769 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5770 \AddToHook{begindocument}{
5771   \ltx@ifpackageloaded{babel}{
5772     \makeatletter
5773     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5774     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5775       \input{problem-ngerman.ldf}
5776     }
5777     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5778       \input{problem-finnish.ldf}
5779     }
5780     \clist_if_in:NnT \l_tmpa_clist {french}{
5781       \input{problem-french.ldf}
5782     }
5783     \clist_if_in:NnT \l_tmpa_clist {russian}{
5784       \input{problem-russian.ldf}
5785     }
5786     \makeatother
5787   }{ }
5788 }
```

40.2 Problems and Solutions

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

```
5789 \keys_define:nn{ problem / problem }{
5790   id      .str_set:x:N = \l__problems_prob_id_str,
5791   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5792   min     .tl_set:N    = \l__problems_prob_min_tl,
5793   title   .tl_set:N    = \l__problems_prob_title_tl,
5794   type    .tl_set:N    = \l__problems_prob_type_tl,
5795   refnum  .int_set:N   = \l__problems_prob_refnum_int
5796 }
5797 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5798 \str_clear:N \l__problems_prob_id_str
5799 \tl_clear:N \l__problems_prob_pts_tl
5800 \tl_clear:N \l__problems_prob_min_tl
5801 \tl_clear:N \l__problems_prob_title_tl
5802 \tl_clear:N \l__problems_prob_type_tl
5803 \int_zero_new:N \l__problems_prob_refnum_int
5804 \keys_set:nn { problem / problem }{ #1 }
5805 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5806   \let\l__problems_prob_refnum_int\undefined
5807 }
5808 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5809 \newcounter{problem}
5810 \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.

```

5811 \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

```

5812 \newcommand\prob@number{
5813   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5814     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5815   }{
5816     \int_if_exist:NTF \l__problems_prob_refnum_int {
5817       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5818     }{
5819       \prob@label\theproblem
5820     }
5821   }
5822 }

```

(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.

```

5823 \newcommand\prob@title[3]{%
5824   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5825     #2 \l__problems_inclprob_title_tl #3
5826   }{
5827     \tl_if_exist:NTF \l__problems_prob_title_tl {
5828       #2 \l__problems_prob_title_tl #3
5829     }{
5830       #1
5831     }
5832   }
5833 }

```

(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.

```
5834 \def\prob@heading{
5835   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5836   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5837 }
```

(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`

```
5838 \newenvironment{sproblem}[1][]{
5839   \__problems_prob_args:n{#1}%\sref@target%
5840   \@in@omtexttrue% we are in a statement (for inline definitions)
5841   \stepcounter{problem}\record@problem
5842   \def\current@section@level{\prob@problem@kw}
5843   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5844     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5845   }{
5846     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5847   }
5848   \str_if_exist:NTF \l__problems_inclprob_id_str {
5849     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5850   }{
5851     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5852   }
5853
5854
5855   \clist_set:No \l_tmpa_clist \sproblemtype
5856   \tl_clear:N \l_tmpa_tl
5857   \clist_map_inline:Nn \l_tmpa_clist {
5858     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5859       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5860     }
5861   }
5862   \tl_if_empty:NTF \l_tmpa_tl {
5863     \__problems_sproblem_start:
5864   }{
5865     \l_tmpa_tl
5866   }
5867   \stex_ref_new_doc_target:n \sproblemid
5868 }{
5869   \clist_set:No \l_tmpa_clist \sproblemtype
5870   \tl_clear:N \l_tmpa_tl
5871   \clist_map_inline:Nn \l_tmpa_clist {
5872     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5873       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5874     }
5875   }
```

```

5875 }
5876 \tl_if_empty:NTF \l_tmpa_tl {
5877   \__problems_sproblem_end:
5878 }{
5879   \l_tmpa_tl
5880 }
5881
5882
5883 \smallskip
5884 }
5885
5886
5887 \cs_new_protected:Nn \__problems_sproblem_start: {
5888   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5889 }
5890 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5891
5892 \newcommand\stexpatchproblem[3]{} {
5893   \str_set:Nx \l_tmpa_str{ #1 }
5894   \str_if_empty:NTF \l_tmpa_str {
5895     \tl_set:Nn \__problems_sproblem_start: { #2 }
5896     \tl_set:Nn \__problems_sproblem_end: { #3 }
5897   }{
5898     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5899     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5900   }
5901 }
5902
5903
5904 \bool_if:NT \c__problems_boxed_bool {
5905   \surroundwithmdframed{problem}
5906 }

```

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

```

5907 \def\record@problem{
5908   \protected@write\@auxout{}
5909   {
5910     \string\@problem{\prob@number}
5911     {
5912       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5913         \l__problems_inclprob_pts_tl
5914       }{
5915         \l__problems_prob_pts_tl
5916       }
5917     }%
5918     {
5919       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5920         \l__problems_inclprob_min_tl
5921       }{
5922         \l__problems_prob_min_tl
5923       }
5924     }
5925   }
5926 }

```

(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).

```
5927 \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.

```
5928 \keys_define:nn { problem / solution }{
5929   id          .str_set_x:N = \l__problems_solution_id_str ,
5930   for         .tl_set:N    = \l__problems_solution_for_tl ,
5931   height      .dim_set:N   = \l__problems_solution_height_dim ,
5932   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5933   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5934   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5935 }
5936 \cs_new_protected:Nn \__problems_solution_args:n {
5937   \str_clear:N \l__problems_solution_id_str
5938   \tl_clear:N \l__problems_solution_for_tl
5939   \tl_clear:N \l__problems_solution_srccite_tl
5940   \clist_clear:N \l__problems_solution_creators_clist
5941   \clist_clear:N \l__problems_solution_contributors_clist
5942   \dim_zero:N \l__problems_solution_height_dim
5943   \keys_set:nn { problem / solution }{ #1 }
5944 }
```

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

```
5945 \newcommand\@startsolution[1][{}]{
5946   \__problems_solution_args:n { #1 }
5947   \@in@omtexttrue% we are in a statement.
5948   \bool_if:NF \c__problems_boxed_bool { \hrule }
5949   \smallskip\noindent
5950   {\textbf\prob@solution@kw : \enspace}
5951   \begin{small}
5952   \def\current@section@level{\prob@solution@kw}
5953   \ignorespacesandpars
5954 }
```

\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.

```
5955 \newcommand\startsolutions{
5956   \specialcomment{solution}{\@startsolution}{
5957     \bool_if:NF \c__problems_boxed_bool {
5958       \hrule\medskip
5959     }
5960     \end{small}%
5961   }
5962   \bool_if:NT \c__problems_boxed_bool {
5963     \surroundwithmdframed{solution}
5964   }
5965 }
```

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

\stopsolutions

```
5966 \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.

```
5967 \ifsolutions
5968   \startsolutions
5969 \else
5970   \stopsolutions
5971 \fi
```

exnote

```
5972 \bool_if:NTF \c__problems_notes_bool {
5973   \newenvironment{exnote}[1][]{
5974     \par\smallskip\hrule\smallskip
5975     \noindent\textbf{\prob@note@kw : }\small
5976   }{
5977     \smallskip\hrule
5978   }
5979 }{
5980   \excludecomment{exnote}
5981 }
```

hint

```
5982 \bool_if:NTF \c__problems_notes_bool {
5983   \newenvironment{hint}[1][]{
5984     \par\smallskip\hrule\smallskip
5985     \noindent\textbf{\prob@hint@kw :~ }\small
5986   }{
5987     \smallskip\hrule
5988   }
5989 \newenvironment{exhint}[1][]{
5990   \par\smallskip\hrule\smallskip
5991   \noindent\textbf{\prob@hint@kw :~ }\small
5992 }{
5993   \smallskip\hrule
5994 }
5995 }{
5996   \excludecomment{hint}
5997   \excludecomment{exhint}
5998 }
```

gnote

```
5999 \bool_if:NTF \c__problems_notes_bool {
6000   \newenvironment{gnote}[1][]{
6001     \par\smallskip\hrule\smallskip
6002     \noindent\textbf{\prob@gnote@kw : }\small
6003   }{
6004     \smallskip\hrule
6005   }
6006 }{
6007   \excludecomment{gnote}
6008 }
```


40.3 Multiple Choice Blocks

```

6009 \newenvironment{mcb}{
6010   \begin{enumerate}
6011 }{
6012   \end{enumerate}
6013 }

```

we define the keys for the mcc macro

```

6014 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6015   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6016     \bool_set_true:N #1
6017   }{
6018     \bool_set_false:N #1
6019   }
6020 }
6021 \keys_define:nn { problem / mcc }{
6022   id          .str_set:N = \l__problems_mcc_id_str ,
6023   feedback    .tl_set:N  = \l__problems_mcc_feedback_tl ,
6024   T           .default:n  = { true } ,
6025   T           .bool_set:N = \l__problems_mcc_t_bool ,
6026   F           .default:n  = { true } ,
6027   F           .bool_set:N = \l__problems_mcc_f_bool ,
6028   Ttext       .code:n     = {
6029     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6030   } ,
6031   Ftext       .code:n     = {
6032     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6033   }
6034 }
6035 \cs_new_protected:Nn \l__problems_mcc_args:n {
6036   \str_clear:N \l__problems_mcc_id_str
6037   \tl_clear:N \l__problems_mcc_feedback_tl
6038   \bool_set_true:N \l__problems_mcc_t_bool
6039   \bool_set_true:N \l__problems_mcc_f_bool
6040   \bool_set_true:N \l__problems_mcc_Ttext_bool
6041   \bool_set_false:N \l__problems_mcc_Ftext_bool
6042   \keys_set:nn { problem / mcc }{ #1 }
6043 }

```

\mcc

```

6044 \newcommand\mcc[2][] {
6045   \l__problems_mcc_args:n{ #1 }
6046   \item #2
6047   \ifsolutions
6048     \\\
6049     \bool_if:NT \l__problems_mcc_t_bool {
6050       % TODO!
6051       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6052     }
6053     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

6054      % TODO!
6055      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6056    }
6057    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6058      !
6059    }{
6060      \l__problems_mcc_feedback_tl
6061    }
6062    \fi
6063  } %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.

```

6064
6065 \keys_define:nn{ problem / inclproblem }{
6066   id      .str_set:N = \l__problems_inclprob_id_str,
6067   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6068   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6069   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6070   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6071   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6072   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6073 }
6074 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6075   \str_clear:N \l__problems_prob_id_str
6076   \tl_clear:N \l__problems_inclprob_pts_tl
6077   \tl_clear:N \l__problems_inclprob_min_tl
6078   \tl_clear:N \l__problems_inclprob_title_tl
6079   \tl_clear:N \l__problems_inclprob_type_tl
6080   \int_zero_new:N \l__problems_inclprob_refnum_int
6081   \str_clear:N \l__problems_inclprob_mhrepos_str
6082   \keys_set:nn { problem / inclproblem }{ #1 }
6083   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6084     \let\l__problems_inclprob_pts_tl\undefined
6085   }
6086   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6087     \let\l__problems_inclprob_min_tl\undefined
6088   }
6089   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6090     \let\l__problems_inclprob_title_tl\undefined
6091   }
6092   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6093     \let\l__problems_inclprob_type_tl\undefined
6094   }
6095   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6096     \let\l__problems_inclprob_refnum_int\undefined
6097   }
6098 }

```

```

6099
6100 \cs_new_protected:Nn \__problems_inclprob_clear: {
6101   \let\l__problems_inclprob_id_str\undefined
6102   \let\l__problems_inclprob_pts_tl\undefined
6103   \let\l__problems_inclprob_min_tl\undefined
6104   \let\l__problems_inclprob_title_tl\undefined
6105   \let\l__problems_inclprob_type_tl\undefined
6106   \let\l__problems_inclprob_refnum_int\undefined
6107   \let\l__problems_inclprob_mhrepos_str\undefined
6108 }
6109 \__problems_inclprob_clear:
6110
6111 \newcommand\includeproblem[2][ ]{
6112   \__problems_inclprob_args:n{ #1 }
6113   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6114     \input{#2}
6115   }{
6116     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6117       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6118     }
6119   }
6120   \__problems_inclprob_clear:
6121 }

```

(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.

```

6122 \AddToHook{enddocument}{
6123   \bool_if:NT \c__problems_pts_bool {
6124     \message{Total:~\arabic{pts}~points}
6125   }
6126   \bool_if:NT \c__problems_min_bool {
6127     \message{Total:~\arabic{min}~minutes}
6128   }
6129 }

```

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

```

6130 \def\pts#1{
6131   \bool_if:NT \c__problems_pts_bool {
6132     \marginpar{#1~\prob@pt@kw}
6133   }
6134 }
6135 \def\min#1{
6136   \bool_if:NT \c__problems_min_bool {
6137     \marginpar{#1~\prob@min@kw}
6138   }
6139 }

```

`\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.

```

6140 \newcounter{pts}
6141 \def\show@pts{
6142   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6143     \bool_if:NT \c__problems_pts_bool {
6144       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6145       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6146     }
6147   }{
6148     \tl_if_exist:NT \l__problems_prob_pts_tl {
6149       \bool_if:NT \c__problems_pts_bool {
6150         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6151         \addtocounter{pts}{\l__problems_prob_pts_tl}
6152       }
6153     }
6154   }
6155 }

```

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

`\show@min`

```

6156 \newcounter{min}
6157 \def\show@min{
6158   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6159     \bool_if:NT \c__problems_min_bool {
6160       \marginpar{\l__problems_inclprob_min_tl\ min}
6161       \addtocounter{min}{\l__problems_inclprob_min_tl}
6162     }
6163   }{
6164     \tl_if_exist:NT \l__problems_prob_min_tl {
6165       \bool_if:NT \c__problems_min_bool {
6166         \marginpar{\l__problems_prob_min_tl\ min}
6167         \addtocounter{min}{\l__problems_prob_min_tl}
6168       }
6169     }
6170   }
6171 }
6172 </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.

```
6173 <@@=hwexam>
6174 <*cls>
6175 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6176 \RequirePackage{l3keys2e,expl-keystr-compatible}
6177 \DeclareOption*{
6178   \PassOptionsToClass{\CurrentOption}{document-structure}
6179   \PassOptionsToPackage{\CurrentOption}{stex}
6180   \PassOptionsToPackage{\CurrentOption}{hwexam}
6181   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6182 }
6183 \ProcessOptions
```

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

```
6184 \LoadClass{document-structure}
6185 \RequirePackage{stex}
6186 \RequirePackage{hwexam}
6187 \RequirePackage{tikzinput}
6188 \RequirePackage{graphicx}
6189 \RequirePackage{a4wide}
6190 \RequirePackage{amssymb}
6191 \RequirePackage{amstext}
6192 \RequirePackage{amsmath}
```

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

```

6193 \newcommand\assig@default@type{\hwexam@assignment@kw}
6194 \def\document@hwexamtype{\assig@default@type}
6195 <@@=document_structure>
6196 \keys_define:nn { document-structure / document }{
6197 id .str_set_x:N = \c_document_structure_document_id_str,
6198 hwexamtype .tl_set:N = \document@hwexamtype
6199 }
6200 <@@=hwexam>
6201 </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.

```
6202 \*package>
6203 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6204 \RequirePackage{l3keys2e,expl-keystr-compat}
6205
6206 \newif\iftest\testfalse
6207 \DeclareOption{test}{\testtrue}
6208 \newif\ifmultiple\multiplefalse
6209 \DeclareOption{multiple}{\multipletrue}
6210 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6211 \ProcessOptions
```

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

```
6212 \RequirePackage{keyval}[1997/11/10]
6213 \RequirePackage{problem}
```

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

```
6214 \newcommand\hwexam@assignment@kw{Assignment}
6215 \newcommand\hwexam@given@kw{Given}
6216 \newcommand\hwexam@due@kw{Due}
6217 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6218 blank~for~extra~space}
6219 \def\hwexam@minutes@kw{minutes}
6220 \newcommand\correction@probs@kw{prob.}
6221 \newcommand\correction@pts@kw{total}
6222 \newcommand\correction@reached@kw{reached}
6223 \newcommand\correction@sum@kw{Sum}
6224 \newcommand\correction@grade@kw{grade}
6225 \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

```

6226 \AddToHook{begindocument}{
6227 \ltx@ifpackageloaded{babel}{
6228 \makeatletter
6229 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6230 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6231 \input{hwexam-ngerman.ldf}
6232 }
6233 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6234 \input{hwexam-finnish.ldf}
6235 }
6236 \clist_if_in:NnT \l_tmpa_clist {french}{
6237 \input{hwexam-french.ldf}
6238 }
6239 \clist_if_in:NnT \l_tmpa_clist {russian}{
6240 \input{hwexam-russian.ldf}
6241 }
6242 \makeatother
6243 }{}
6244 }
6245

```

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.

```

6246 \newcounter{assignment}
6247 \numberproblemsin{assignment}
6248 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6249 \keys_define:nn { hwexam / assignment } {
6250 id .str_set:N = \l__hwexam_assign_id_str,
6251 number .int_set:N = \l__hwexam_assign_number_int,
6252 title .tl_set:N = \l__hwexam_assign_title_tl,
6253 type .tl_set:N = \l__hwexam_assign_type_tl,
6254 given .tl_set:N = \l__hwexam_assign_given_tl,
6255 due .tl_set:N = \l__hwexam_assign_due_tl,
6256 loadmodules .code:n = {
6257 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6258 }
6259 }
6260 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6261 \str_clear:N \l__hwexam_assign_id_str
6262 \int_set:Nn \l__hwexam_assign_number_int {-1}
6263 \tl_clear:N \l__hwexam_assign_title_tl
6264 \tl_clear:N \l__hwexam_assign_type_tl
6265 \tl_clear:N \l__hwexam_assign_given_tl
6266 \tl_clear:N \l__hwexam_assign_due_tl
6267 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```



```

6268 \keys_set:nn { hwexam / assignment }{ #1 }
6269 }

```

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.

```

6270 \newcommand\given@due[2]{
6271 \bool_lazy_all:nF {
6272 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6273 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6274 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6275 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6276 }{ #1 }
6277
6278 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6279 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6280 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6281 }
6282 }{
6283 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6284 }
6285
6286 \bool_lazy_or:nnF {
6287 \bool_lazy_and_p:nn {
6288 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6289 }{
6290 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6291 }
6292 }{
6293 \bool_lazy_and_p:nn {
6294 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6295 }{
6296 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6297 }
6298 }{ ,~ }
6299
6300 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6301 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6302 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6303 }
6304 }{
6305 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6306 }
6307
6308 \bool_lazy_all:nF {
6309 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6310 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6311 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6312 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6313 }{ #2 }
6314 }

```

`\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.

```

6315 \newcommand\assignment@title[3]{
6316 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6317 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6318 #1
6319 }{
6320 #2\l__hwexam_assign_title_tl#3
6321 }
6322 }{
6323 #2\l__hwexam_inclasssign_title_tl#3
6324 }
6325 }

```

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

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

```

6326 \newcommand\assignment@number{
6327 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6328 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6329 \arabic{assignment}
6330 } {
6331 \int_use:N \l__hwexam_assign_number_int
6332 }
6333 }{
6334 \int_use:N \l__hwexam_inclasssign_number_int
6335 }
6336 }

```

(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.

```

6337 \newenvironment{assignment}[1][ ]{
6338 \__hwexam_assignment_args:n { #1 }
6339 %\sref@target
6340 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6341 \global\stepcounter{assignment}
6342 }{
6343 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6344 }
6345 \setcounter{problem}{0}
6346 \def\current@section@level{\document@hwexamtype}
6347 %\sref@label@id{\document@hwexamtype \thesection}
6348 \begin{@assignment}
6349 }{
6350 \end{@assignment}
6351 }

```

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

```

6352 \def\ass@title{
6353 \protect\document@hwexamtype~\arabic{assignment}
6354 \assignment@title{}\{;\}{} -- \given@due{}\}{}
6355 }
6356 \ifmultiple
6357 \newenvironment{@assignment}{
6358 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6359 \begin{omgroup}[loadmodules]{\ass@title}
6360 }{
6361 \begin{omgroup}{\ass@title}
6362 }
6363 }{
6364 \end{omgroup}
6365 }

```

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

```

6366 \else
6367 \newenvironment{@assignment}{
6368 \begin{center}\bf
6369 \Large@title\strut\\
6370 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
6371 \large\given@due{--;\}{}{;\}{}
6372 \end{center}
6373 }{}
6374 \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.

```

6375 \keys_define:nn { hwexam / inclassignment } {
6376 %id .str_set_x:N = \l__hwexam_assign_id_str,
6377 number .int_set:N = \l__hwexam_inclassign_number_int,
6378 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6379 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6380 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6381 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6382 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6383 }
6384 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6385 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6386 \tl_clear:N \l__hwexam_inclassign_title_tl
6387 \tl_clear:N \l__hwexam_inclassign_type_tl
6388 \tl_clear:N \l__hwexam_inclassign_given_tl
6389 \tl_clear:N \l__hwexam_inclassign_due_tl
6390 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6391 \keys_set:nn { hwexam / inclassignment }{ #1 }
6392 }
6393 \__hwexam_inclassignment_args:n {}
6394
6395 \newcommand\inputassignment[2][{}]{

```

```

6396 \_hwexam_inclassnment_args:n { #1 }
6397 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6398 \input{#2}
6399 }{
6400 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6401 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6402 }
6403 }
6404 \_hwexam_inclassnment_args:n {}
6405 }
6406 \newcommand\includeassignment[2][ ]{
6407 \newpage
6408 \inputassignment[#1]{#2}
6409 }

```

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

42.4 Typesetting Exams

\quizheading

```

6410 \ExplSyntaxOff
6411 \newcommand\quizheading[1]{%
6412 \def\@tas{#1}%
6413 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6414 \ifx\@tas\@empty\else%
6415 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6416 \fi%
6417 }
6418 \ExplSyntaxOn

```

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

\testheading

```

6419
6420 \def\hwexamheader{\input{hwexam-default.header}}
6421
6422 \def\hwexamminutes{
6423 \tl_if_empty:NTF \testheading@duration {
6424 {\testheading@min}~\hwexam@minutes@kw
6425 }{
6426 \testheading@duration
6427 }
6428 }
6429
6430 \keys_define:nn { hwexam / testheading } {
6431 min .tl_set:N = \testheading@min,
6432 duration .tl_set:N = \testheading@duration,
6433 reqpts .tl_set:N = \testheading@reqpts,
6434 tools .tl_set:N = \testheading@tools
6435 }
6436 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6437 \tl_clear:N \testheading@min
6438 \tl_clear:N \testheading@duration

```

```

6439 \tl_clear:N \testheading@reqpts
6440 \tl_clear:N \testheading@tools
6441 \keys_set:nn { hwexam / testheading }{ #1 }
6442 }
6443 \newenvironment{testheading}[1][ ]{
6444   \_hwexam_testheading_args:n{ #1 }
6445   \newcount\check@time\check@time=\testheading@min
6446   \advance\check@time by -\theassignment@totalmin
6447   \newif\if@bonuspoints
6448   \tl_if_empty:NTF \testheading@reqpts {
6449     \@bonuspointsfalse
6450   }{
6451     \newcount\bonus@pts
6452     \bonus@pts=\theassignment@totalpts
6453     \advance\bonus@pts by -\testheading@reqpts
6454     \edef\bonus@pts{\the\bonus@pts}
6455     \@bonuspointstrue
6456   }
6457   \edef\check@time{\the\check@time}
6458
6459   \makeatletter\hwexamheader\makeatother
6460 }{
6461   \newpage
6462 }

```

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

\testspace

```

6463 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6464 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6465 \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.

```

6466 <@=problems>
6467 \renewcommand\@problem[3]{
6468   \stepcounter{assignment@probs}
6469   \def\__problemspts{#2}
6470   \ifx\__problemspts\@empty\else
6471     \addtocounter{assignment@totalpts}{#2}
6472   \fi
6473   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6474   \xdef\correction@probs{\correction@probs & #1}%
6475   \xdef\correction@pts{\correction@pts & #2}
6476   \xdef\correction@reached{\correction@reached &}

```

```

6477 }
6478 <@@=hwexam>

```

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

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

```

6479 \newcounter{assignment@probs}
6480 \newcounter{assignment@totalpts}
6481 \newcounter{assignment@totalmin}
6482 \def\correction@probs{\correction@probs@kw}
6483 \def\correction@pts{\correction@pts@kw}
6484 \def\correction@reached{\correction@reached@kw}
6485 \stepcounter{assignment@probs}
6486 \newcommand\correction@table{
6487 \resizebox{\textwidth}{!}{%
6488 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6489 &\multicolumn{\theassignment@probs}{c|}|%|
6490 {\footnotesize\correction@forgrading@kw} &\\ \hline
6491 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6492 \correction@pts & \theassignment@totalpts & \\ \hline
6493 \correction@reached & & \[.7cm]\hline
6494 \end{tabular}}
6495 </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}

```