

The sTeX3 Package *

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

2022-02-17

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 2022-02-17)

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	99
28.1	The module environment	102
28.2	Invoking modules	108
29	STeX-Module Inheritance Implementation	110
29.1	SMS Mode	110
29.2	Inheritance	113
30	STeX-Symbols Implementation	118
30.1	Symbol Declarations	118
30.2	Notations	125
31	STeX-Terms Implementation	136
31.1	Symbol Invocations	136
31.2	Terms	139
31.3	Notation Components	146
32	STeX-Structural Features Implementation	149
32.1	Imports with modification	149
32.2	The feature environment	156
32.3	Features	158
33	STeX-Statements Implementation	163
33.1	Definitions	163
33.2	Assertions	168
33.3	Examples	171
33.4	Logical Paragraphs	173

34 The Implementation	179
34.1 Package Options	179
34.2 Proofs	179
34.3 Justifications	185
35 \LaTeX-Others Implementation	187
36 \LaTeX-Metatheory Implementation	188
37 Tikzinput Implementation	191
38 document-structure.sty Implementation	193
38.1 The document-structure Class	193
38.2 Class Options	193
38.3 Beefing up the <code>document</code> environment	194
38.4 Implementation: document-structure Package	194
38.5 Package Options	194
38.6 Document Structure	196
38.7 Front and Backmatter	199
38.8 Global Variables	201
39 NotesSlides – Implementation	202
39.1 Class and Package Options	202
39.2 Notes and Slides	204
39.3 Header and Footer Lines	208
39.4 Frame Images	209
39.5 Colors and Highlighting	210
39.6 Sectioning	211
39.7 Excursions	213
40 The Implementation	215
40.1 Package Options	215
40.2 Problems and Solutions	216
40.3 Multiple Choice Blocks	222
40.4 Including Problems	223
40.5 Reporting Metadata	224
41 Implementation: The hwexam Class	226
41.1 Class Options	226
42 Implementation: The hwexam Package	228
42.1 Package Options	228
42.2 Assignments	229
42.3 Including Assignments	232
42.4 Typesetting Exams	233
42.5 Leftovers	235

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

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}
\renamedecl[name=universe]{universe}{runiverse}
\renamedecl[name=plus]{operation}{rplus}
\renamedecl[name=zero]{unit}{rzero}
\renamedecl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedecl[name=times]{operation}{rtimes}
\renamedecl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\compl}
Test:  $\$ \rtimes a \{ \rplus \ c \{ \rtimes \ de \} \}$ 
\end{smodule}
```

Module 5:

Test: $a \circ a$

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 $\text{\texttt{STEX}}$ Metatheory)

Chapter 6

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

Chapter 7

Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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}][ $\$a$ ][\comp{and}][ $\$b$ ]
```

$a * b$ 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  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

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

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

Example 11

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

$\textcolor{teal}{\text{Multiplication}}$ (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 ’s operator precedence should be smaller than B ’s argument precedences.

For example:

Module 9:

Example 14

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

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

8.1.2 Archives and Imports

Namespaces

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

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

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 9

sTeX-Basics

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

debug ($\langle log\text{-}prefix \rangle *$) 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 sTeX	Both print this sTeX logo.
\backslash stex	

\backslash stex_debug:nn	\backslash 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 stex_add_to_sms:n	Adds the provided code to the .sms-file of the document.
--------------------------------	--

\backslash if@latexml	L ^A T _E X ₂ e and L ^A T _E X ₃ conditionals for L ^A T _E X _{ML} .
\backslash latexml_if_p:	
\backslash latexml_if:T	
\backslash latexml_if:F	
\backslash 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](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?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dummyvar>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <file://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\lbracket #1 \comp\rbracket } }
$\foobar a{b,c,d,e,f}g$ 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 $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[{$\displaystyle
\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$}
\end{smodule}
```

Module 26: $\langle a|_{[b;c;d;e;f]}^g \rangle$ and $\langle a|_{[b;c]}^g \rangle$ and $\langle a|_{[b]^c} \rangle$

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

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

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

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

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

Test 16

```
\begin{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 its sister package `statements`.

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

Proof:	We prove that $\sum_{i=1}^n 2i - 1 = n^2$ by induction over n	
P.1	For the induction we have to consider the following cases:	
P.1.1	$n = 1$: then we compute $1 = 1^2$	□
P.1.1	$n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$	□
P.1.1	$n > 1$:	
P.1.1.1	Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.	
P.1.1.1	We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.	
P.1.1.1	We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum	
P.1.1.1	Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.	
P.1.1.1	We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion.	□
P.1.1	We have considered all the cases, so we have proven the assertion.	□

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

18.2.4 Proof Structure

subproof	The pfcases environment is used to mark up a subproof. This environment takes an optional KeyVal argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the proof environment). The method key can be used to give the name of the proof method executed to make this subproof.
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 spfcases environments that mark up the cases one by one.
spfcases	The content of a pfcases environment are a sequence of case proofs marked up in the pfcases 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 pfcases environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcases 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 do, or what we have achieved so far. As such, it cannot be the target of a \premise .

18.2.5 Proof End Markers

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

`\sproofend`

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

`\sProofEndSymbol`

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

18.2.6 Configuration of the Presentation

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

EdN:8

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

The `document-structure` package is part of the $\S\TeX$ collection, a version of \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
<code>id</code>	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>creators</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>contributors</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>short</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>loadmodules</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <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 \LaTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

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

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

21.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[⟨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 \LaTeX ML generate the correct reference.
`\STRcopy`
`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.¹⁰

21.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the \LaTeX preamble of the course notes file. `\setSGvar{⟨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 \LaTeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

22.2.1 Package Options

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

- | | |
|---------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 22.2.2). |

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

22.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.⁴

⚠ Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else L^AT_EX becomes confused and throws error messages that are difficult to decipher.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between

¹¹EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

⁴MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.

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

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nparagraph` environment is just an `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

22.2.3 Header and Footer Lines of the Slides

The default logo provided by the `notesslides` package is the \TeX logo it can be customized using `\setslidelogo{<logo name>}`.

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

22.2.4 Frame Images

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

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

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

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

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


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

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

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

```
\mhframeimage{baz/foobar}
```

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

```
Problem 0.1 (Fitting Elephants)
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

gnote The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

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

Example 8: A generated test heading.

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 sTeXlogo:
\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_EX_ML, R_US_TE_X, p_DF_LA_TE_X).

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

```

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
301 \protected\def\ignorespacesandpars{
302   \begingroup\catcode13=10\relax
303   \@ifnextchar\par{
304     \endgroup\expandafter\ignorespacesandpars\@gobble
305   }{
306     \endgroup
307   }
308 }
309
310
311 </package>

```

Chapter 26

STEX -MathHub Implementation

```
312 <*package>
313
314 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
315
316 <@@=stex_path>
    Warnings and error messages
317 \msg_new:nnn{stex}{error/norepository}{
318   No~archive~#1~found~in~#2
319 }
320 \msg_new:nnn{stex}{error/notinarchive}{
321   Not~currently~in~an~archive,~but~\detokenize{#1}~
322   needs~one!
323 }
324 \msg_new:nnn{stex}{error/nofile}{
325   \detokenize{#1}~could~not~find~file~#2
326 }
327 \msg_new:nnn{stex}{error/twofiles}{
328   \detokenize{#1}~found~two~candidates~for~#2
329 }
```

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
330 \cs_new_protected:Nn \stex_path_from_string:Nn {
331   \str_set:Nx \l_tmpa_str { #2 }
332   \str_if_empty:NTF \l_tmpa_str {
333     \seq_clear:N #1
334   }{
335     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
336     \sys_if_platform_windows:T{
337       \seq_clear:N \l_tmpa_tl
```

```

338     \seq_map_inline:Nn #1 {
339       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
340       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
341     }
342     \seq_set_eq:NN #1 \l_tmpa_tl
343   }
344   \stex_path_canonicalize:N #1
345 }
346 }
347 \cs_generate_variant:Nn \stex_path_from_string:Nn
348 { 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`

```

349 \cs_new_protected:Nn \stex_path_to_string:NN {
350   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
351 }
352
353 \cs_new:Nn \stex_path_to_string:N {
354   \seq_use:Nn #1 /
355 }

```

(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`

```

356 \str_const:Nn \c__stex_path_dot_str {.}
357 \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.

```

358 \cs_new_protected:Nn \stex_path_canonicalize:N {
359   \seq_if_empty:NF #1 {
360     \seq_clear:N \l_tmpa_seq
361     \seq_get_left:NN #1 \l_tmpa_tl
362     \str_if_empty:NT \l_tmpa_tl {
363       \seq_put_right:Nn \l_tmpa_seq {}
364     }
365     \seq_map_inline:Nn #1 {
366       \str_set:Nn \l_tmpa_tl { ##1 }
367       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
368         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
369           \seq_if_empty:NNTF \l_tmpa_seq {
370             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
371               \c__stex_path_up_str
372             }
373           }{
374             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
375             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
376               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
377                 \c__stex_path_up_str
378               }

```

```

379         }{
380         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
381         }
382     }
383     ){
384     \str_if_empty:NF \l_tmpa_tl {
385     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
386     }
387     }
388 }
389 }
390 \seq_gset_eq:NN #1 \l_tmpa_seq
391 }
392 }

```

(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}`

```

393 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
394   \seq_if_empty:NTF #1 {
395     \prg_return_false:
396   }{
397     \seq_get_left:NN #1 \l_tmpa_tl
398     \str_if_empty:NTF \l_tmpa_tl {
399       \prg_return_true:
400     }{
401       \prg_return_false:
402     }
403   }
404 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

405 \str_new:N\l_stex_kpsewhich_return_str
406 \cs_new_protected:Nn \stex_kpsewhich:n {
407   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
408   \exp_args:NNo \str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
409   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
410 }

```

(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`

```

411 \sys_if_platform_windows:TF{
412   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
413 }{
414   \stex_kpsewhich:n{-var-value~PWD}
415 }
416

```



```

417 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
418 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
419 \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

```

420 <@@=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

```

421 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

422 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
423 \stex_path_from_string:Nn \c_stex_mainfile_seq
424 \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`.

```

425 \seq_gclear_new:N\g_stex_currentfile_seq
426 \cs_new_protected:Nn \stex_filestack_push:n {
427   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
428   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
429     \stex_path_from_string:Nn\g_stex_currentfile_seq{
430       \c_stex_pwd_str/#1
431     }
432   }
433   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
434   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
435 }
436 \cs_new_protected:Nn \stex_filestack_pop: {
437   \seq_if_empty:NF\g_stex_files_stack{
438     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
439   }
440   \seq_if_empty:NTF\g_stex_files_stack{
441     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
442   }{
443     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
444     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
445   }
446 }
447

```

```

448 \AddToHook{file/before}{
449   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
450 }
451 \AddToHook{file/after}{
452   \stex_filestack_pop:
453 }

```

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

26.4 MathHub Repositories

```

454 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
455 \str_if_empty:NTF\mathhub{
456   \stex_kpsewhich:n{-var-value~MATHHUB}
457   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
458
459   \str_if_empty:NTF\c_stex_mathhub_str{
460     \msg_warning:nn{stex}{warning/nomathhub}
461   }{
462     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
464   }
465 }{
466   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
467   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
468     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
469       \c_stex_pwd_str/\mathhub
470     }
471   }
472   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
473   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
474 }

```

(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
475 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
476   \str_set:Nx \l_tmpa_str { #1 }
477   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
478     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
479     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
480     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
481     \__stex_mathhub_find_manifest:N \l_tmpa_seq
482     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
483       \msg_error:nnxx{stex}{error/norepository}{#1}{
484         \stex_path_to_string:N \c_stex_mathhub_str
485       }
486     } {
487       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
488     }
489   }
490 }

```

(End definition for _stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

491 \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:

```

492 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
493   \seq_set_eq:NN\l_tmpa_seq #1
494   \bool_set_true:N\l_tmpa_bool
495   \bool_while_do:Nn \l_tmpa_bool {
496     \seq_if_empty:NTF \l_tmpa_seq {
497       \bool_set_false:N\l_tmpa_bool
498     }{
499       \file_if_exist:nTF{
500         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
501       }{
502         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
503         \bool_set_false:N\l_tmpa_bool
504       }{
505         \file_if_exist:nTF{
506           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
507         }{
508           \seq_put_right:Nn\l_tmpa_seq{META-INF}
509           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
510           \bool_set_false:N\l_tmpa_bool
511         }{
512           \file_if_exist:nTF{
513             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
514           }{
515             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
516             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
517             \bool_set_false:N\l_tmpa_bool
518           }{
519             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
520           }
521         }
522       }
523     }
524   }
525   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
526 }

```

(End definition for _stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior File variable used for MANIFEST-files

527 \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:

```

528 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
529   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
530   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
531   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
532     \str_set:Nn \l_tmpa_str {##1}
533     \exp_args:NNo \seq_set_split:Nnn
534       \l_tmpb_seq \c_colon_str \l_tmpa_str
535     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
536       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
537         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
538       }
539       \exp_args:No \str_case:nnTF \l_tmpa_tl {
540         {id} {
541           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
542             { id } \l_tmpb_tl
543         }
544         {narration-base} {
545           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
546             { narr } \l_tmpb_tl
547         }
548         {url-base} {
549           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
550             { docurl } \l_tmpb_tl
551         }
552         {source-base} {
553           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
554             { ns } \l_tmpb_tl
555         }
556         {ns} {
557           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
558             { ns } \l_tmpb_tl
559         }
560         {dependencies} {
561           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
562             { deps } \l_tmpb_tl
563         }
564       }{}{}
565     }{}
566   }
567   \ior_close:N \c__stex_mathhub_manifest_ior
568 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

569 \cs_new_protected:Nn \stex_set_current_repository:n {
570   \stex_require_repository:n { #1 }
571   \prop_set_eq:Nc \l_stex_current_repository_prop {
572     c_stex_mathhub_#1_manifest_prop
573   }
574 }

```

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

`\stex_require_repository:n`

```

575 \cs_new_protected:Nn \stex_require_repository:n {
576   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
577     \stex_debug:nn{mathhub}{Opening~archive:~#1}
578     \__stex_mathhub_do_manifest:n { #1 }
579     \exp_args:Nx \stex_add_to_sms:n {
580       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
581         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
582         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
583         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
584         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
585       }
586     }
587   }
588 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

589 %\prop_new:N \l_stex_current_repository_prop
590
591 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
592 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
593   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
594 } {
595   \__stex_mathhub_parse_manifest:n { main }
596   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
597   \l_tmpa_str
598   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
599   \c_stex_mathhub_main_manifest_prop
600   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
601   \stex_debug:nn{mathhub}{Current~repository:~
602     \prop_item:Nn \l_stex_current_repository_prop {id}
603   }
604 }

```

(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.

```

605 \cs_new_protected:Nn \stex_in_repository:nn {
606   \str_set:Nx \l_tmpa_str { #1 }
607   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
608   \str_if_empty:NTF \l_tmpa_str {
609     \prop_if_exist:NTF \l_stex_current_repository_prop {
610       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
611       \exp_args:Ne \l_tmpa_cs{
612         \prop_item:Nn \l_stex_current_repository_prop { id }
613       }
614     }{
615       \l_tmpa_cs{}
616     }
617   }{
618     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

619 \stex_require_repository:n \l_tmpa_str
620 \str_set:Nx \l_tmpa_str { #1 }
621 \exp_args:Nne \use:nn {
622   \stex_set_current_repository:n \l_tmpa_str
623   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
624 }{
625   \stex_debug:nn{mathhub}{switching~back~to:~
626     \prop_if_exist:NTF \l_stex_current_repository_prop {
627       \prop_item:Nn \l_stex_current_repository_prop { id }::~
628       \meaning\l_stex_current_repository_prop
629     }{
630       no~repository
631     }
632   }
633   \prop_if_exist:NTF \l_stex_current_repository_prop {
634     \stex_set_current_repository:n {
635       \prop_item:Nn \l_stex_current_repository_prop { id }
636     }
637   }{
638     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
639   }
640 }
641 }
642 }

```

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

```

\inputref
\stex_inputref:nn 643 \newif \ifinputref \inputreffalse
\mhinput\stex_mhinput:nn 644
645 \cs_new_protected:Nn \stex_mhinput:nn {
646   \stex_in_repository:nn {#1} {
647     \ifinputref
648       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
649     \else
650       \inputreftrue
651       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
652     \inputreffalse
653   \fi
654 }
655 }
656 \NewDocumentCommand \mhinput { 0{} m}{
657   \stex_mhinput:nn{ #1 }{ #2 }
658 }
659
660 \cs_new_protected:Nn \stex_inputref:nn {
661   \stex_in_repository:nn {#1} {
662     \bool_lazy_any:nTF {
663       {\rustex_if_p:} {\latexml_if_p:}
664     } {
665       \str_clear:N \l_tmpa_str
666       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
667         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
668       }

```

```

669     \stex_annotate_invisible:nnn{inputref}{
670       \l_tmpa_str / #2
671     }{}
672   }{
673     \begingroup
674     \inputreftrue
675     \input{ \c_stex_mathhub_str / ##1 / source / #2 }
676   \endgroup
677   }
678 }
679 }
680
681 \NewDocumentCommand \inputref { 0{ } m}{
682   \stex_inputref:nn{ #1 }{ #2 }
683 }
684
685 \cs_new_protected:Nn \stex_mhbibresource:nn {
686   \stex_in_repository:nn {#1} {
687     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
688   }
689 }
690 \newcommand\addmhbibresource[2][]{
691   \stex_mhbibresource:nn{ #1 }{ #2 }
692 }

```

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

`\mhpath`

```

693 \def \mhpath #1 #2 {
694   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
695     \c_stex_mathhub_str /
696     \prop_item:Nn \l_stex_current_repository_prop { id }
697     / source / #2
698   }{
699     \c_stex_mathhub_str / #1 / source / #2
700   }
701 }

```

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

`\libinput`

```

702 \cs_new_protected:Npn \libinput #1 {
703   \prop_if_exist:NF \l_stex_current_repository_prop {
704     \msg_error:nnn{stex}{error/notinarchive}\libinput
705   }
706   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
707     \msg_error:nnn{stex}{error/notinarchive}\libinput
708   }
709   \tl_clear:N \l__stex_mathhub_libinput_files_seq
710   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
711   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
712
713   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
714     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}

```

```

715 \IfFileExists{ \l_tmpa_str }{
716   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
717 }{}
718 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
719 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
720 }
721
722 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
723 \IfFileExists{ \l_tmpa_str }{
724   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
725 }{}
726
727 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
728   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
729 }{
730   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
731     \input{ ##1 }
732   }
733 }
734 }

```

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

`\libusepackage`

```

735 \NewDocumentCommand \libusepackage {0{ } m} {
736   \prop_if_exist:NF \l_stex_current_repository_prop {
737     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
738   }
739   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
740     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
741   }
742   \tl_clear:N \l__stex_mathhub_libinput_files_seq
743   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
744   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
745
746   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
747     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
748     \IfFileExists{ \l_tmpa_str }{
749       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
750     }{}
751     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
752     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
753   }
754
755   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
756   \IfFileExists{ \l_tmpa_str }{
757     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
758   }{}
759
760   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
761     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
762   }{
763     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
764       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {

```



```

765         \usepackage[#1]{ ##1 }
766     }
767 }{
768     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
769 }
770 }
771 }

```

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

```

772
773 \AddToHook{begindocument}{
774 \ltx@ifpackageloaded{graphicx}{
775     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
776     \newcommand\mhgraphics[2][]{\%
777         \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
778         \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
779     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
780 }{}
781 \ltx@ifpackageloaded{listings}{
782     \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
783     \newcommand\lstinputmhlisting[2][]{\%
784         \def\lst@mhrepos{}\setkeys{lst}{#1}%
785         \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
786     \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
787 }{}
788 }
789
790
791 \</package>

```

Chapter 27

STEX -References Implementation

```
792 <*package>
793
794 %%%%%%%%%%% references.dtx %%%%%%%%%%%
795
796 %\RequirePackage{hyperref}
797 %\RequirePackage{cleveref}
798 <@@=stex_refs>
799
800 Warnings and error messages
801
802 \iow_new:N \c__stex_refs_refs_iow
803 \AddToHook{begindocument}{
804   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
805 }
806 \AddToHook{enddocument}{
807   \iow_close:N \c__stex_refs_refs_iow
808 }
809
810 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
811
812 \NewDocumentCommand \STEXreftitle { m } {
813   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
814 }
```

27.1 Document URIs and URLs

```
813
814 \str_new:N \l_stex_current_docns_str
815
816 \cs_new_protected:Nn \stex_get_document_uri: {
817   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
818   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
819   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
820   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
821   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
```

```

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

```

```

876     }
877   }{
878     \bool_set_true:N \l_tmpa_bool
879     \bool_while_do:Nn \l_tmpa_bool {
880       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
881       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
882         {source} { \bool_set_false:N \l_tmpa_bool }
883       }{}{
884         \seq_if_empty:NT \l_tmpa_seq {
885           \bool_set_false:N \l_tmpa_bool
886         }
887       }
888     }
889
890     \seq_if_empty:NTF \l_tmpa_seq {
891       \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
892     }{
893       \str_set:Nx \l_stex_current_docurl_str {
894         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
895       }
896     }
897   }
898 }

```

27.2 Setting Reference Targets

```

899 \str_const:Nn \c__stex_refs_url_str{URL}
900 \str_const:Nn \c__stex_refs_ref_str{REF}
901 \str_new:N \l__stex_refs_curr_label_str
902 % @currentlabel -> number
903 % @currentlabelname -> title
904 % @currentHref -> name.number <- id of some kind
905 % \theH# -> \arabic{section}
906 % \the# -> number
907 % \hyper@makecurrent{#}
908 \int_new:N \l__stex_refs_unnamed_counter_int
909 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
910   \stex_get_document_uri:
911   \str_clear:N \l__stex_refs_curr_label_str
912   \str_set:Nx \l_tmpa_str { #1 }
913   \str_if_empty:NT \l_tmpa_str {
914     \int_incr:N \l__stex_refs_unnamed_counter_int
915     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
916   }
917   \str_set:Nx \l__stex_refs_curr_label_str {
918     \l_stex_current_docns_str?\l_tmpa_str
919   }
920   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
921     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
922   }
923   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
924     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
925   }
926   \stex_if_smsmode:TF {

```

```

927 \stex_get_document_url:
928 \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
929 \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
930 }{
931 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
932 \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
933 \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
934 \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
935 }
936 }
937
938 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
939 \str_set:Nn \l_tmpa_str {#1?#2}
940 \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
941 \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
942 \seq_new:c {g__stex_refs_labels_#2_seq}
943 }
944 \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
945 \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
946 }
947 }
948
949 \AtEndDocument{
950 \def\stexauxadddocref#1 #2 {}{}}
951 }
952
953 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
954 \stex_if_smsmode:TF {
955 \str_if_exist:cF{sref_sym_#1_type}{
956 \stex_get_document_url:
957 \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
958 \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
959 }
960 }{
961 \str_if_empty:NF \l__stex_refs_curr_label_str {
962 \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
963 \immediate\write\@auxout{
964 \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
965 \l__stex_refs_curr_label_str
966 }
967 }
968 }
969 }
970 }

```

27.3 Using References

```

971 \str_new:N \l__stex_refs_indocument_str
972 \keys_define:nn { stex / sref } {
973 linktext .tl_set:N = \l__stex_refs_linktext_tl ,
974 fallback .tl_set:N = \l__stex_refs_fallback_tl ,
975 pre .tl_set:N = \l__stex_refs_pre_tl ,
976 post .tl_set:N = \l__stex_refs_post_tl ,
977 %indoc .str_set:x:N = \l__stex_refs_repo_str ,

```

```

978 }
979
980
981
982 \cs_new_protected:Nn \__stex_refs_args:n {
983   \tl_clear:N \l__stex_refs_linktext_tl
984   \tl_clear:N \l__stex_refs_fallback_tl
985   \tl_clear:N \l__stex_refs_pre_tl
986   \tl_clear:N \l__stex_refs_post_tl
987   \str_clear:N \l__stex_refs_repo_str
988   \keys_set:nn { stex / sref } { #1 }
989 }
990
991 \NewDocumentCommand \sref { 0{} m}{
992   \__stex_refs_args:n { #1 }
993   \str_if_empty:NTF \l__stex_refs_indocument_str {
994     \str_set:Nx \l_tmpa_str { #2 }
995     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
996     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
997       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
998         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
999           \str_clear:N \l_tmpa_str
1000         }
1001       }{
1002         \str_clear:N \l_tmpa_str
1003       }
1004     }{
1005       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1006       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1007       \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1008       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1009         \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1010         \str_clear:N \l_tmpa_str
1011         \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1012           \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1013             \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1014           }{
1015             \seq_map_break:n {
1016               \str_set:Nn \l_tmpa_str { ##1 }
1017             }
1018           }
1019         }
1020       }{
1021         \str_clear:N \l_tmpa_str
1022       }
1023     }
1024     \str_if_empty:NTF \l_tmpa_str {
1025       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1026     }{
1027       \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1028         \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1029           \cs_if_exist:cTF{autoref}{
1030             \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1031           }{

```

```

1032         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1033     }
1034 }{
1035     \ltx@ifpackageloaded{hyperref}{
1036         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1037     }{
1038         \l__stex_refs_linktext_tl
1039     }
1040 }
1041 }{
1042     \ltx@ifpackageloaded{hyperref}{
1043         \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1044     }{
1045         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1046     }
1047 }
1048 }
1049 }{
1050     % TODO
1051 }
1052 }
1053
1054 \NewDocumentCommand \srefsym { 0{ } m }{
1055     \stex_get_symbol:n { #2 }
1056     \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1057 }
1058
1059 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1060     \str_if_exist:cTF {sref_sym_#2 _label_str }{
1061         \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1062     }{
1063         \__stex_refs_args:n { #1 }
1064         \str_if_empty:NTF \l__stex_refs_indocument_str {
1065             \tl_if_exist:cTF{sref_sym_#2 _type}{
1066                 % doc uri in \l_tmpb_str
1067                 \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1068                 \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1069                     % reference
1070                     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1071                         \cs_if_exist:cTF{autoref}{
1072                             \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1073                         }{
1074                             \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1075                         }
1076                     }{
1077                         \ltx@ifpackageloaded{hyperref}{
1078                             \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1079                         }{
1080                             \l__stex_refs_linktext_tl
1081                         }
1082                     }
1083                 }{
1084                     % URL
1085                     \ltx@ifpackageloaded{hyperref}{

```

```

1086         \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1087     }{
1088         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1089     }
1090 }
1091 }{
1092     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1093 }
1094 }{
1095     % TODO
1096 }
1097 }
1098 }
1099
1100 \cs_new_protected:Npn \srefsymuri #1 #2 {
1101     \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1102 }
1103
1104 \end{package}

```


Chapter 28

STEX -Modules Implementation

```
1105 <*package>
1106
1107 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1108
1109 <@@=stex_modules>
1110
1111   Warnings and error messages
1112 \msg_new:nnn{stex}{error/unknownmodule}{
1113   No~module~#1~found
1114 }
1115 \msg_new:nnn{stex}{error/syntax}{
1116   Syntax~error:~#1
1117 }
1118 \msg_new:nnn{stex}{error/siglanguage}{
1119   Module~#1~declares~signature~#2,~but~does~not~
1120   declare~its~language
1121 }
1122 \msg_new:nnn{stex}{warning/deprecated}{
1123   #1~is~deprecated;~please~use~#2~instead!
1124 }
1125 \msg_new:nnn{stex}{error/conflictingmodules}{
1126   Conflicting~imports~for~module~#1
1127 }
```

`\l_stex_current_module_str` The current module:

```
1127 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 26.)

`\l_stex_all_modules_seq` Stores all available modules

```
1128 \seq_new:N \l_stex_all_modules_seq
```

(End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1129 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1130   \str_if_empty:NTF \l_stex_current_module_str
1131   \prg_return_false: \prg_return_true:
1132 }

```

(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
1133 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1134   \prop_if_exist:cTF { c_stex_module_#1_prop }
1135   \prg_return_true: \prg_return_false:
1136 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1137 \cs_new_protected:Nn \stex_add_to_current_module:n {
1138   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1139 }
1140 \cs_new_protected:Npn \STEXexport {
1141   \beginngroup
1142   \newlinechar=-1\relax
1143   \endlinechar=-1\relax
1144   %\catcode'\ = 9\relax
1145   \expandafter\endgroup\STEXexport:n
1146 }
1147 \cs_new_protected:Nn \STEXexport:n {
1148   \ignorespaces #1
1149   \stex_add_to_current_module:n { \ignorespaces #1 }
1150   \stex_smsmode_do:
1151 }
1152 \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
1153 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1154   \str_set:Nx \l_tmpa_str { #1 }
1155   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1156 }
1157
1158 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1159 % \str_set:Nx \l_tmpa_str { #1 }
1160 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1161 %}

```

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

```

\stex_collect_imports:n
1162 \cs_new_protected:Nn \stex_collect_imports:n {
1163   \seq_clear:N \l_stex_collect_imports_seq
1164   \__stex_modules_collect_imports:n {#1}

```

```

1165 }
1166 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1167   \seq_map_inline:cn {c_stex_module_#1_imports} {
1168     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1169       \__stex_modules_collect_imports:n { ##1 }
1170     }
1171   }
1172   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1173     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1174   }
1175 }

```

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

`\stex_add_import_to_current_module:n`

```

1176 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1177   \str_set:Nx \l_tmpa_str { #1 }
1178   \exp_args:Nno
1179   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1180     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1181   }
1182 }

```

(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).

```

1183 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1184   \str_set:Nx \l_tmpa_str { #1 }
1185   \seq_set_eq:NN \l_tmpa_seq #2
1186   % split off file extension
1187   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1188   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1189   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1190   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1191
1192   \bool_set_true:N \l_tmpa_bool
1193   \bool_while_do:Nn \l_tmpa_bool {
1194     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1195     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1196       {source} { \bool_set_false:N \l_tmpa_bool }
1197     }{}{
1198       \seq_if_empty:NT \l_tmpa_seq {
1199         \bool_set_false:N \l_tmpa_bool
1200       }
1201     }
1202   }
1203
1204   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1205   \str_if_empty:NTF \l_stex_modules_subpath_str {
1206     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1207   }{
1208     \str_set:Nx \l_stex_modules_ns_str {
1209       \l_tmpa_str/\l_stex_modules_subpath_str

```

```

1210     }
1211   }
1212 }

```

(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
1213 \str_new:N \l_stex_modules_ns_str
1214 \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.

```

1215 \cs_new_protected:Nn \stex_modules_current_namespace: {
1216   \str_clear:N \l_stex_modules_subpath_str
1217   \prop_if_exist:NTF \l_stex_current_repository_prop {
1218     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1219     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1220   }{
1221     % split off file extension
1222     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1223     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1224     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1225     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1226     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1227     \str_set:Nx \l_stex_modules_ns_str {
1228       file:/\stex_path_to_string:N \l_tmpa_seq
1229     }
1230   }
1231 }

```

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

28.1 The module environment

module arguments:

```

1232 \keys_define:nn { stex / module } {
1233   title      .tl_set:N      = \smodulename ,
1234   type       .str_set_x:N   = \smodulename ,
1235   id         .str_set_x:N   = \smodulename ,
1236   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1237   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1238   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1239   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1240   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1241   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1242   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1243   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1244 }
1245
1246 \cs_new_protected:Nn \__stex_modules_args:n {

```

```

1247 \str_clear:N \smoduletitle
1248 \str_clear:N \smoduletype
1249 \str_clear:N \smoduleid
1250 \str_clear:N \l_stex_module_ns_str
1251 \str_clear:N \l_stex_module_deprecate_str
1252 \str_clear:N \l_stex_module_lang_str
1253 \str_clear:N \l_stex_module_sig_str
1254 \str_clear:N \l_stex_module_creators_str
1255 \str_clear:N \l_stex_module_contributors_str
1256 \str_clear:N \l_stex_module_meta_str
1257 \str_clear:N \l_stex_module_srccite_str
1258 \keys_set:nn { stex / module } { #1 }
1259 }
1260
1261 % module parameters here? In the body?
1262

```

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

```

1263 \cs_new_protected:Nn \stex_module_setup:nn {
1264   \str_set:Nx \l_stex_module_name_str { #2 }
1265   \__stex_modules_args:n { #1 }
1266
1267   First, we set up the name and namespace of the module.
1268   Are we in a nested module?
1269
1270   \stex_if_in_module:TF {
1271     % Nested module
1272     \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1273       { ns } \l_stex_module_ns_str
1274     \str_set:Nx \l_stex_module_name_str {
1275       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1276         { name } / \l_stex_module_name_str
1277     }
1278   }{
1279     % not nested:
1280     \str_if_empty:NT \l_stex_module_ns_str {
1281       \stex_modules_current_namespace:
1282       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1283       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1284         / {\l_stex_module_ns_str}
1285       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1286       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1287         \str_set:Nx \l_stex_module_ns_str {
1288           \stex_path_to_string:N \l_tmpa_seq
1289         }
1290       }
1291     }
1292   }
1293 }

```

Next, we determine the language of the module:

```

1289 \str_if_empty:NT \l_stex_module_lang_str {
1290   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1291   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1292   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1293   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>

```

```

1294 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1295 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1296   inferred~from~file~name}
1297 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1298 }
1299 }
1300
1301 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1302 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1303 \l_tmpa_str {
1304 \ltx@ifpackageloaded{babel}{
1305 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1306 }{}
1307 } {
1308 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1309 }
1310 }}

```

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

```

1311 \str_if_empty:NNTF \l_stex_module_sig_str {
1312 \exp_args:Nnx \prop_gset_from_keyval:cn {
1313   c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1314 } {
1315   name      = \l_stex_module_name_str ,
1316   ns        = \l_stex_module_ns_str ,
1317   file      = \exp_not:o { \g_stex_currentfile_seq } ,
1318   lang      = \l_stex_module_lang_str ,
1319   sig       = \l_stex_module_sig_str ,
1320   deprecate = \l_stex_module_deprecate_str ,
1321   meta      = \l_stex_module_meta_str
1322 }
1323 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1324 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1325 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1326 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1327 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1328 \str_if_empty:NT \l_stex_module_meta_str {
1329 \str_set:Nx \l_stex_module_meta_str {
1330   \c_stex_metatheory_ns_str ? Metatheory
1331 }
1332 }
1333 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1334 \bool_set_true:N \l_stex_in_meta_bool
1335 \exp_args:Nx \stex_add_to_current_module:n {
1336   \bool_set_true:N \l_stex_in_meta_bool
1337   \stex_activate_module:n {\l_stex_module_meta_str}
1338   \bool_set_false:N \l_stex_in_meta_bool
1339 }
1340 \stex_activate_module:n {\l_stex_module_meta_str}
1341 \bool_set_false:N \l_stex_in_meta_bool
1342 }

```

```

1343 }{
1344   \str_if_empty:NT \l_stex_module_lang_str {
1345     \msg_error:nnxx{stex}{error/siglanguage}{
1346       \l_stex_module_ns_str?\l_stex_module_name_str
1347     }\l_stex_module_sig_str}
1348   }
1349
1350   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1351   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1352   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1353   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1354   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1355   \str_set:Nx \l_tmpa_str {
1356     \stex_path_to_string:N \l_tmpa_seq /
1357     \l_tmpa_str . \l_stex_module_sig_str .tex
1358   }
1359   \IfFileExists \l_tmpa_str {
1360     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1361       \str_clear:N \l_stex_current_module_str
1362       \seq_clear:N \l_stex_all_modules_seq
1363       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1364     }
1365   }{
1366     \msg_error:nnxx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1367   }
1368   \stex_if_smsmode:F {
1369     \stex_activate_module:n {
1370       \l_stex_module_ns_str ? \l_stex_module_name_str
1371     }
1372   }
1373   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1374 }
1375 \str_if_empty:NF \l_stex_module_deprecate_str {
1376   \msg_warning:nnxx{stex}{warning/deprecated}{
1377     Module~\l_stex_current_module_str
1378   }{
1379     \l_stex_module_deprecate_str
1380   }
1381 }
1382 }

```

(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}
1383 \int_new:N \l_stex_module_group_depth_int
1384 \cs_new_protected:Nn \__stex_modules_begin_module: {
1385   \stex_reactivate_macro:N \STEXexport
1386   \stex_reactivate_macro:N \importmodule
1387   \stex_reactivate_macro:N \symdecl
1388   \stex_reactivate_macro:N \notation
1389   \stex_reactivate_macro:N \symdef
1390 }

```

```

1391 \stex_debug:nn{modules}{
1392   New~module:\\
1393   Namespace::~\l_stex_module_ns_str\\
1394   Name::~\l_stex_module_name_str\\
1395   Language::~\l_stex_module_lang_str\\
1396   Signature::~\l_stex_module_sig_str\\
1397   Metatheory::~\l_stex_module_meta_str\\
1398   File::~\stex_path_to_string:N \g_stex_currentfile_seq
1399 }
1400
1401 \seq_put_right:Nx \l_stex_all_modules_seq {
1402   \l_stex_module_ns_str ? \l_stex_module_name_str
1403 }
1404
1405 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1406 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1407
1408
1409 \stex_if_smsmode:F{
1410   \begin{stex_annotate_env} {theory} {
1411     \l_stex_module_ns_str ? \l_stex_module_name_str
1412   }
1413
1414   \stex_annotate_invisible:nnn{header}{} {
1415     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1416     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1417     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1418       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1419     }
1420     \str_if_empty:NF \smodulotype {
1421       \stex_annotate:nnn{type}{\smodulotype}{}
1422     }
1423   }
1424 }
1425 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1426 % TODO: Inherit metatheory for nested modules?
1427 }
1428 \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}

```

1429 \cs_new_protected:Nn \_stex_modules_end_module: {
1430 % \str_set:Nx \l_tmpa_str {
1431 %   c_stex_module_
1432 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1433 %   \prop_item:Nn \l_stex_current_module_prop { name }
1434 %   _prop
1435 % }
1436 %%^A \prop_new:c { \l_tmpa_str }
1437 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1438 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1439 }

```

(End definition for _stex_modules_end_module:.)

smodule The core environment, with no header

```

1440 \iffalse \begin{stex_annotate_env} \fi %^A make syntax highlighting work again
1441 \NewDocumentEnvironment { smodule } { 0{} m } {
1442   \stex_module_setup:nn{#1}{#2}
1443   \par
1444   \stex_if_smsmode:F{
1445     \tl_clear:N \l_tmpa_tl
1446     \clist_map_inline:Nn \smoduletype {
1447       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1448         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1449       }
1450     }
1451     \tl_if_empty:NTF \l_tmpa_tl {
1452       \__stex_modules_smodule_start:
1453     }{
1454       \l_tmpa_tl
1455     }
1456   }
1457   \__stex_modules_begin_module:
1458   \str_if_empty:NF \smoduleid {
1459     \stex_ref_new_doc_target:n \smoduleid
1460   }
1461   \stex_smsmode_do:
1462 } {
1463   \__stex_modules_end_module:
1464   \stex_if_smsmode:TF {
1465     % \exp_args:Nx \stex_add_to_sms:n {
1466     %   \prop_gset_from_keyval:cn {
1467     %     c_stex_module_
1468     %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1469     %     \prop_item:Nn \l_stex_current_module_prop { name }
1470     %     _prop
1471     %   } {
1472     %     name      = \prop_item:cn { \l_tmpa_str } { name } ,
1473     %     ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1474     %     file      = \prop_item:cn { \l_tmpa_str } { file } ,
1475     %     lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1476     %     sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1477     %     meta      = \prop_item:cn { \l_tmpa_str } { meta }
1478     %   }
1479     % }
1480   }{
1481     \end{stex_annotate_env}
1482     \clist_set:No \l_tmpa_clist \smoduletype
1483     \tl_clear:N \l_tmpa_tl
1484     \clist_map_inline:Nn \l_tmpa_clist {
1485       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1486         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1487       }
1488     }
1489     \tl_if_empty:NTF \l_tmpa_tl {
1490       \__stex_modules_smodule_end:
1491     }{
1492       \l_tmpa_tl

```

```

1493     }
1494   }
1495 }
1496
1497 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1498 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1499
1500 \newcommand\stexpatchmodule[3] [] {
1501   \str_set:Nx \l_tmpa_str{ #1 }
1502   \str_if_empty:NTF \l_tmpa_str {
1503     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1504     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1505   }{
1506     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1507     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1508   }
1509 }
1510

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1511 \NewDocumentCommand \STEXModule { m } {
1512   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1513   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1514   \tl_set:Nn \l_tmpa_tl {
1515     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1516   }
1517   \seq_map_inline:Nn \l_stex_all_modules_seq {
1518     \str_set:Nn \l_tmpb_str { ##1 }
1519     \str_if_eq:eeT { \l_tmpa_str } {
1520       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1521     } {
1522       \seq_map_break:n {
1523         \tl_set:Nn \l_tmpa_tl {
1524           \stex_invoke_module:n { ##1 }
1525         }
1526       }
1527     }
1528   }
1529   \l_tmpa_tl
1530 }
1531
1532 \cs_new_protected:Nn \stex_invoke_module:n {
1533   \stex_debug:nn{modules}{Invoking~module~#1}
1534   \peek_charcode_remove:NTF ! {
1535     \__stex_modules_invoke_uri:nN { #1 }
1536   } {
1537     \peek_charcode_remove:NTF ? {
1538       \__stex_modules_invoke_symbol:nn { #1 }
1539     } {
1540       \msg_error:nnx{stex}{error/syntax}{
1541         ?~or~!~expected~after~

```

```

1542     \c_backslash_str STEXModule{#1}
1543   }
1544 }
1545 }
1546 }
1547
1548 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1549   \str_set:Nn #2 { #1 }
1550 }
1551
1552 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1553   \stex_invoke_symbol:n{#1?#2}
1554 }

```

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

`\stex_activate_module:n`

```

1555 \bool_new:N \l_stex_in_meta_bool
1556 \bool_set_false:N \l_stex_in_meta_bool
1557 \cs_new_protected:Nn \stex_activate_module:n {
1558   \stex_debug:nn{modules}{Activating~module~#1}
1559   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1560     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1561   }
1562   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1563     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1564     \use:c{ c_stex_module_#1_code }
1565   }
1566 }

```

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

```

1567 </package>

```

Chapter 29

STEX -Module Inheritance Implementation

```
1568 <*package>
1569
1570 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1571
```

29.1 SMS Mode

```
1572 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1573 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1574 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1575 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1576
1577 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1578   \makeatletter
1579   \makeatother
1580   \ExplSyntaxOn
1581   \ExplSyntaxOff
1582   \rustexBREAK
1583 }
1584
1585 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1586   \symdef
1587   \importmodule
1588   \notation
1589   \symdecl
1590   \STEXexport
1591   \inlineass
1592   \inlinedef
1593   \inlineex
1594   \endinput
1595   \setnotation
```

```

1596 \copynotation
1597 }
1598
1599 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1600 \tl_to_str:n {
1601   smodule,
1602   copymodule,
1603   interpretmodule
1604   sdefinition,
1605   sexample,
1606   sassertion,
1607   sparagraph
1608 }
1609 }

```

(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`

```

1610 \bool_new:N \g__stex_smsmode_bool
1611 \bool_set_false:N \g__stex_smsmode_bool
1612 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1613   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1614 }

```

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

`\stex_in_smsmode:nn`

```

1615 \cs_new_protected:Nn \stex_in_smsmode:nn {
1616   \vbox_set:Nn \l_tmpa_box {
1617     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1618     \bool_gset_true:N \g__stex_smsmode_bool
1619     #2
1620     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1621   }
1622   \box_clear:N \l_tmpa_box
1623 }
1624
1625 \quark_new:N \q__stex_smsmode_break
1626
1627 %\ior_new:N \c__stex_smsmode_ior
1628 %\tl_new:N \l__stex_smsmode_filecontent_tl
1629 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1630   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1631   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1632   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1633     % \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1634   % }
1635   % \ior_close:N \c__stex_smsmode_ior
1636   \stex_filestack_push:n{#1}
1637   \stex_in_smsmode:nn{#1} {
1638     #2
1639     \everyeof{\q__stex_smsmode_break\noexpand}
1640     \expandafter\expandafter\expandafter
1641     \stex_smsmode_do:

```

```

1642 \csname @ @ input\endcsname "#1"\relax
1643 %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1644 }
1645 \stex_filestack_pop:
1646 }

```

(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:

```

1647 \cs_new_protected:Npn \stex_smsmode_do: {
1648   \stex_if_smsmode:T {
1649     \__stex_smsmode_do:w
1650   }
1651 }
1652 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1653   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1654     \expandafter\if\expandafter\relax\noexpand#1
1655     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1656   \else\expandafter\__stex_smsmode_do:w\fi
1657 }{
1658   \__stex_smsmode_do:w % #1
1659 }
1660 }
1661 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1662   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1663     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1664       #1\__stex_smsmode_do:w
1665     }{
1666       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1667         #1
1668       }{
1669         \cs_if_eq:NNTF \begin #1 {
1670           \__stex_smsmode_check_begin:n
1671         }{
1672           \cs_if_eq:NNTF \end #1 {
1673             \__stex_smsmode_check_end:n
1674           }{
1675             \__stex_smsmode_do:w
1676           }
1677         }
1678       }
1679     }
1680   }
1681 }
1682
1683 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1684   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1685     \begin{#1}
1686   }{
1687     \__stex_smsmode_do:w
1688   }
1689 }
1690 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1691 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1692   \end{#1}\__stex_smsmode_do:w
1693 }{
1694   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1695 }
1696 }

```

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

29.2 Inheritance

```

1697 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1698 \cs_new_protected:Nn \stex_import_module_uri:nn {
1699   \str_set:Nx \l_stex_import_archive_str { #1 }
1700   \str_set:Nn \l_stex_import_path_str { #2 }
1701
1702   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1703   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1704   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1705
1706   \stex_modules_current_namespace:
1707   \bool_lazy_all:nTF {
1708     {\str_if_empty_p:N \l_stex_import_archive_str}
1709     {\str_if_empty_p:N \l_stex_import_path_str}
1710     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1711   }{
1712     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1713     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1714   }{
1715     \str_if_empty:NT \l_stex_import_archive_str {
1716       \prop_if_exist:NT \l_stex_current_repository_prop {
1717         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1718       }
1719     }
1720     \str_if_empty:NTF \l_stex_import_archive_str {
1721       \str_if_empty:NF \l_stex_import_path_str {
1722         \str_set:Nx \l_stex_import_ns_str {
1723           \l_stex_module_ns_str / \l_stex_import_path_str
1724         }
1725       }
1726     }{
1727       \stex_require_repository:n \l_stex_import_archive_str
1728       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1729       \l_stex_import_ns_str
1730       \str_if_empty:NF \l_stex_import_path_str {
1731         \str_set:Nx \l_stex_import_ns_str {
1732           \l_stex_import_ns_str / \l_stex_import_path_str
1733         }
1734       }
1735     }
1736   }
1737 }

```

(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 1738 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1739 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1740 \str_new:N \l_stex_import_path_str
1741 \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:nmmn {<ns>} {<archive-ID>} {<path>} {<name>}
1742 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1743   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1744
1745     % archive
1746     \str_set:Nx \l_tmpa_str { #2 }
1747     \str_if_empty:NTF \l_tmpa_str {
1748       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1749     } {
1750       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1751       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1752       \seq_put_right:Nn \l_tmpa_seq { source }
1753     }
1754
1755     % path
1756     \str_set:Nx \l_tmpb_str { #3 }
1757     \str_if_empty:NTF \l_tmpb_str {
1758       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1759
1760       \ltx@ifpackageloaded{babel} {
1761         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1762           { \languagename } \l_tmpb_str {
1763           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1764         }
1765       } {
1766         \str_clear:N \l_tmpb_str
1767       }
1768
1769       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1770       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1771         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1772       }{
1773         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1774         \IfFileExists{ \l_tmpa_str.tex }{
1775           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1776         }{
1777           % try english as default
1778           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1779           \IfFileExists{ \l_tmpa_str.en.tex }{
1780             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1781           }{
1782             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1783           }
1784         }

```



```

1785     }
1786
1787   } {
1788     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1789     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1790
1791     \ltx@ifpackageloaded{babel} {
1792       \exp_args:Nnx \prop_get:NnNF \c_stex_language_abbrevs_prop
1793       { \language } \l_tmpb_str {
1794         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1795       }
1796     } {
1797       \str_clear:N \l_tmpb_str
1798     }
1799
1800     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1801
1802     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1803     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1804       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1805     }{
1806       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1807       \IfFileExists{ \l_tmpa_str/#4.tex }{
1808         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1809       }{
1810         % try english as default
1811         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1812         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1813           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1814         }{
1815           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1816           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1817             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1818           }{
1819             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1820             \IfFileExists{ \l_tmpa_str.tex }{
1821               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1822             }{
1823               % try english as default
1824               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1825               \IfFileExists{ \l_tmpa_str.en.tex }{
1826                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1827               }{
1828                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1829               }
1830             }
1831           }
1832         }
1833       }
1834     }
1835   }
1836
1837   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1838     \seq_clear:N \l_stex_all_modules_seq

```

```

1839     \str_clear:N \l_stex_current_module_str
1840     \str_set:Nx \l_tmpb_str { #2 }
1841     \str_if_empty:NF \l_tmpb_str {
1842         \stex_set_current_repository:n { #2 }
1843     }
1844     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1845 }
1846
1847 \stex_if_module_exists:nF { #1 ? #4 } {
1848     \msg_error:nnx{stex}{error/unknownmodule}{
1849         #1?#4~(in~file~\g__stex_importmodule_file_str)
1850     }
1851 }
1852 }
1853 \stex_activate_module:n { #1 ? #4 }
1854 }

```

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

`\importmodule`

```

1855 \NewDocumentCommand \importmodule { 0{} m } {
1856     \stex_import_module_uri:nn { #1 } { #2 }
1857     \stex_debug:nn{modules}{Importing~module:~
1858         \l_stex_import_ns_str ? \l_stex_import_name_str
1859     }
1860     \stex_if_smsmode:F {
1861         \stex_import_require_module:nnnn
1862         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1863         { \l_stex_import_path_str } { \l_stex_import_name_str }
1864         \stex_annotate_invisible:nnn
1865         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1866     }
1867     \exp_args:Nx \stex_add_to_current_module:n {
1868         \stex_import_require_module:nnnn
1869         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1870         { \l_stex_import_path_str } { \l_stex_import_name_str }
1871     }
1872     \exp_args:Nx \stex_add_import_to_current_module:n {
1873         \l_stex_import_ns_str ? \l_stex_import_name_str
1874     }
1875     \stex_smsmode_do:
1876     \ignorespacesandpars
1877 }
1878 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1879 \NewDocumentCommand \usemodule { 0{} m } {
1880     \stex_if_smsmode:F {
1881         \stex_import_module_uri:nn { #1 } { #2 }
1882         \stex_import_require_module:nnnn
1883         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1884         { \l_stex_import_path_str } { \l_stex_import_name_str }
1885         \stex_annotate_invisible:nnn

```

```

1886     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1887   }
1888   \stex_smsmode_do:
1889   \ignorespacesandpars
1890 }

```

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

```

1891 </package>

```

Chapter 30

STEX -Symbols Implementation

```
1892 <*package>
1893
1894 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1895
```

Warnings and error messages

```
1896
```

30.1 Symbol Declarations

```
1897 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1898 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1899 \NewDocumentCommand \STEXsymbol { m } {
1900   \stex_get_symbol:n { #1 }
1901   \exp_args:No
1902   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1903 }
```

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

symdecl arguments:

```
1904 \keys_define:nn { stex / symdecl } {
1905   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1906   local     .bool_set:N  = \l_stex_symdecl_local_bool ,
1907   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1908   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1909   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1910   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1911   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1912   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1913   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
```

```

1914   assoc      .choices:nn   =
1915             {bin,binl,binr,pre,conj,pwconj}
1916             {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
1917   }
1918
1919   \bool_new:N \l_stex_symdecl_make_macro_bool
1920
1921   \cs_new_protected:Nn \__stex_symdecl_args:n {
1922     \str_clear:N \l_stex_symdecl_name_str
1923     \str_clear:N \l_stex_symdecl_args_str
1924     \str_clear:N \l_stex_symdecl_deprecate_str
1925     \str_clear:N \l_stex_symdecl_assoc_type_str
1926     \bool_set_false:N \l_stex_symdecl_local_bool
1927     \tl_clear:N \l_stex_symdecl_type_tl
1928     \tl_clear:N \l_stex_symdecl_definiens_tl
1929
1930     \keys_set:nn { stex / symdecl } { #1 }
1931   }

```

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

```

1932
1933   \NewDocumentCommand \symdecl { s O{} m } {
1934     \__stex_symdecl_args:n { #2 }
1935     \IfBooleanTF #1 {
1936       \bool_set_false:N \l_stex_symdecl_make_macro_bool
1937     } {
1938       \bool_set_true:N \l_stex_symdecl_make_macro_bool
1939     }
1940     \stex_symdecl_do:n { #3 }
1941     \stex_smsmode_do:
1942   }
1943   \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1944   \cs_new_protected:Nn \stex_symdecl_do:n {
1945     \stex_if_in_module:F {
1946       % TODO throw error? some default namespace?
1947     }
1948
1949     \str_if_empty:NT \l_stex_symdecl_name_str {
1950       \str_set:Nx \l_stex_symdecl_name_str { #1 }
1951     }
1952
1953     \prop_if_exist:cT { l_stex_symdecl_
1954       \l_stex_current_module_str ?
1955       \l_stex_symdecl_name_str
1956     }_prop
1957   }{
1958     % TODO throw error (beware of circular dependencies)
1959   }
1960

```

```

1961 \prop_clear:N \l_tmpa_prop
1962 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1963 \seq_clear:N \l_tmpa_seq
1964 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1965 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1966
1967 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1968   \str_if_empty:NF \l_stex_module_deprecate_str {
1969     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1970   }
1971 }
1972 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1973
1974 \exp_args:No \stex_add_constant_to_current_module:n {
1975   \l_stex_symdecl_name_str
1976 }
1977
1978 % arity/args
1979 \int_zero:N \l_tmpb_int
1980
1981 \bool_set_true:N \l_tmpa_bool
1982 \str_map_inline:Nn \l_stex_symdecl_args_str {
1983   \token_case_meaning:NnF ##1 {
1984     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1985     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1986     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1987     {\tl_to_str:n a} {
1988       \bool_set_false:N \l_tmpa_bool
1989       \int_incr:N \l_tmpb_int
1990     }
1991     {\tl_to_str:n B} {
1992       \bool_set_false:N \l_tmpa_bool
1993       \int_incr:N \l_tmpb_int
1994     }
1995   }{
1996     \msg_set:nnn{stex}{error/wrongargs}{
1997       args~value~in~symbol~declaration~for~
1998       \l_stex_current_module_str ?
1999       \l_stex_symdecl_name_str ~
2000       needs~to~be~
2001       i,~a,~b~or~B,~but~##1~given
2002     }
2003     \msg_error:nn{stex}{error/wrongargs}
2004   }
2005 }
2006 \bool_if:NTF \l_tmpa_bool {
2007   % possibly numeric
2008   \str_if_empty:NTF \l_stex_symdecl_args_str {
2009     \prop_put:Nnn \l_tmpa_prop { args } {}
2010     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2011   }{
2012     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2013     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2014     \str_clear:N \l_tmpa_str

```

```

2015     \int_step_inline:nn \l_tmpa_int {
2016         \str_put_right:Nn \l_tmpa_str i
2017     }
2018     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2019 }
2020 } {
2021     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2022     \prop_put:Nnx \l_tmpa_prop { arity }
2023         { \str_count:N \l_stex_symdecl_args_str }
2024 }
2025 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2026
2027
2028 % semantic macro
2029
2030 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2031     \exp_args:Nx \stex_do_aftergroup:n {
2032         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2033             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2034         }}
2035     }
2036
2037     \bool_if:NF \l_stex_symdecl_local_bool {
2038         \exp_args:Nx \stex_add_to_current_module:n {
2039             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2040                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041             } }
2042         }
2043     }
2044 }
2045
2046 % add to all symbols
2047
2048 \bool_if:NF \l_stex_symdecl_local_bool {
2049     \exp_args:Nx \stex_add_to_current_module:n {
2050         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2051             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2052         }
2053     }
2054 %     \exp_args:Nx \stex_add_field_to_current_module:n {
2055 %         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2056 %     }
2057 }
2058
2059 \stex_debug:nn{symbols}{New~symbol:~
2060     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2061     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2062     Args:~\prop_item:Nn \l_tmpa_prop { args }
2063 }
2064
2065 % circular dependencies require this:
2066
2067 \prop_if_exist:cF {
2068     l_stex_symdecl_

```

```

2069     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2070     _prop
2071 } {
2072     \prop_set_eq:cN {
2073         l_stex_symdecl_
2074         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2075         _prop
2076     } \l_tmpa_prop
2077 }
2078
2079 \seq_clear:c {
2080     l_stex_symdecl_
2081     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2082     _notations
2083 }
2084
2085 \bool_if:NF \l_stex_symdecl_local_bool {
2086     \exp_args:Nx
2087     \stex_add_to_current_module:n {
2088         \seq_clear:c {
2089             l_stex_symdecl_
2090             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2091             _notations
2092         }
2093         \prop_set_from_keyval:cn {
2094             l_stex_symdecl_
2095             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2096             _prop
2097         } {
2098             name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2099             module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2100             type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2101             args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2102             arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2103             assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2104         }
2105     }
2106 }
2107
2108 \stex_if_smsmode:TF {
2109     \bool_if:NF \l_stex_symdecl_local_bool {
2110 %         \exp_args:Nx \stex_add_to_sms:n {
2111 %             \prop_set_from_keyval:cn {
2112 %                 l_stex_symdecl_
2113 %                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2114 %                 _prop
2115 %             } {
2116 %                 name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2117 %                 module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2118 %                 local     = \prop_item:Nn \l_tmpa_prop { local }      ,
2119 %                 type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2120 %                 args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2121 %                 arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2122 %                 assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,

```



```

2123 %      }
2124 %      \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2125 %          \l_stex_current_module_str ? \l_stex_symdecl_name_str
2126 %      }
2127 %  }
2128 }
2129 }{
2130   \exp_args:Nx \stex_do_aftergroup:n {
2131     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2132       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2133     }
2134   }
2135   \stex_if_do_html:T {
2136     \stex_annotate_invisible:nnn {symdecl} {
2137       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2138     } {
2139       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2140       \stex_annotate_invisible:nnn{args}{}{
2141         \prop_item:Nn \l_tmpa_prop { args }
2142       }
2143       \stex_annotate_invisible:nnn{macroname}{#1}{}
2144       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2145         \stex_annotate_invisible:nnn{definiens}{}
2146         {\l_stex_symdecl_definiens_tl$}
2147       }
2148     }
2149   }
2150 }
2151 }

```

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

`\stex_get_symbol:n`

```

2152 \str_new:N \l_stex_get_symbol_uri_str
2153
2154 \cs_new_protected:Nn \stex_get_symbol:n {
2155   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2156     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2157   }{
2158     % argument is a string
2159     % is it a command name?
2160     \cs_if_exist:cTF { #1 }{
2161       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2162       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2163       \str_if_empty:NTF \l_tmpa_str {
2164         \exp_args:Nx \cs_if_eq:NNTF {
2165           \tl_head:N \l_tmpa_tl
2166         } \stex_invoke_symbol:n {
2167           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2168         }{
2169           \__stex_symdecl_get_symbol_from_string:n { #1 }
2170         }
2171       } {
2172         \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2173     }
2174   }{
2175     % argument is not a command name
2176     \l_stex_symdecl_get_symbol_from_string:n { #1 }
2177     % \l_stex_all_symbols_seq
2178   }
2179 }
2180 \str_if_eq:eeF {
2181   \prop_item:cn {
2182     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2183   }{ deprecate }
2184 }{}{
2185   \msg_warning:nxxx{stex}{warning/deprecated}{
2186     Symbol~\l_stex_get_symbol_uri_str
2187   }{
2188     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2189   }
2190 }
2191 }
2192
2193 \cs_new_protected:Nn \l_stex_symdecl_get_symbol_from_string:n {
2194   \str_set:Nn \l_tmpa_str { #1 }
2195   \bool_set_false:N \l_tmpa_bool
2196   \stex_if_in_module:T {
2197     \exp_args:Nno \seq_if_in:cnT {c_stex_module\l_stex_current_module_str _constants} { \l_
2198       \bool_set_true:N \l_tmpa_bool
2199       \str_set:Nx \l_stex_get_symbol_uri_str {
2200         \l_stex_current_module_str ? #1
2201       }
2202     }
2203   }
2204   \bool_if:NF \l_tmpa_bool {
2205     \tl_set:Nn \l_tmpa_tl {
2206       \msg_set:nnn{stex}{error/unknownsymbol}{
2207         No~symbol~#1~found!
2208       }
2209       \msg_error:nn{stex}{error/unknownsymbol}
2210     }
2211     \str_set:Nn \l_tmpa_str { #1 }
2212     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2213     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2214       \str_set:Nn \l_tmpb_str { ##1 }
2215       \str_if_eq:eeT { \l_tmpa_str } {
2216         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2217       } {
2218         \seq_map_break:n {
2219           \tl_set:Nn \l_tmpa_tl {
2220             \str_set:Nn \l_stex_get_symbol_uri_str {
2221               ##1
2222             }
2223           }
2224         }
2225       }
2226     }

```

```

2227     \l_tmpa_tl
2228   }
2229 }
2230
2231 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2232   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2233     { \tl_tail:N \l_tmpa_tl }
2234   \tl_if_single:NTF \l_tmpa_tl {
2235     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2236       \exp_after:wN \str_set:Nn \exp_after:wN
2237         \l_stex_get_symbol_uri_str \l_tmpa_tl
2238     }{
2239       % TODO
2240       % tail is not a single group
2241     }
2242   }{
2243     % TODO
2244     % tail is not a single group
2245   }
2246 }

```

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

30.2 Notations

```

2247 <@@=stex_notation>
2248
2249 notation arguments:
2250 \keys_define:nn { stex / notation } {
2251   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2252   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2253   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2254   op        .tl_set:N   = \l__stex_notation_op_tl ,
2255   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2256   primary   .default:n   = {true} ,
2257   unknown   .code:n      = \str_set:Nx
2258     \l__stex_notation_variant_str \l_keys_key_str
2259 }
2260
2261 \cs_new_protected:Nn \stex_notation_args:n {
2262   \str_clear:N \l__stex_notation_lang_str
2263   \str_clear:N \l__stex_notation_variant_str
2264   \str_clear:N \l__stex_notation_prec_str
2265   \tl_clear:N \l__stex_notation_op_tl
2266   \bool_set_false:N \l__stex_notation_primary_bool
2267
2268   \keys_set:nn { stex / notation } { #1 }
2269 }
2270
2271 \notation
2272
2273 \NewDocumentCommand \notation { 0{} m } {
2274   \stex_notation_args:n { #1 }
2275   \tl_clear:N \l_stex_symdecl_definiens_tl
2276   \stex_get_symbol:n { #2 }

```

```

2272 \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2273 }
2274 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

2275 \seq_new:N \l__stex_notation_precedences_seq
2276 \tl_new:N \l__stex_notation_opprec_tl
2277 \int_new:N \l__stex_notation_currarg_int
2278
2279 \cs_new_protected:Nn \stex_notation_do:nn {
2280   \let\l_stex_current_symbol_str\relax
2281   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2282   \seq_clear:N \l__stex_notation_precedences_seq
2283   \tl_clear:N \l__stex_notation_opprec_tl
2284   \prop_get:cnN {
2285     l_stex_symdecl_ #1 _prop
2286   } { args } \l__stex_notation_args_str
2287
2288   % precedences
2289   \prop_get:cnN {
2290     l_stex_symdecl_ #1 _prop
2291   } { arity } \l__stex_notation_arity_str
2292   \str_if_empty:NTF \l__stex_notation_prec_str {
2293     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2294       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2295     }{
2296       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2297     }
2298   } {
2299     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2300       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2301       \int_step_inline:nn { \l__stex_notation_arity_str } {
2302         \exp_args:NNo
2303         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2304       }
2305     }{
2306       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2307       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2308         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2309         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2310           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2311             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2312           \seq_map_inline:Nn \l_tmpa_seq {
2313             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2314           }
2315         }
2316       }{
2317         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2318           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2319         }{
2320           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2321         }

```

```

2322     }
2323   }
2324 }
2325
2326 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2327 \int_step_inline:nn { \l__stex_notation_arity_str } {
2328   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2329     \exp_args:NNo
2330     \seq_put_right:No \l__stex_notation_precedences_seq {
2331       \l__stex_notation_opprec_tl
2332     }
2333   }
2334 }
2335
2336 \tl_clear:N \l__stex_notation_dummyargs_tl
2337
2338 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2339   \exp_args:NNe
2340   \cs_set:Npn \l__stex_notation_macrocode_cs {
2341     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2342     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2343     { \l__stex_notation_opprec_tl }
2344     { \exp_not:n { #2 } }
2345   }
2346   \__stex_notation_final:
2347 }{
2348   \str_if_in:NnTF \l__stex_notation_args_str b {
2349     \exp_args:Nne \use:nn
2350     {
2351       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2352       \cs_set:Npn \l__stex_notation_arity_str } { {
2353         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2354         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2355         { \l__stex_notation_opprec_tl }
2356         { \exp_not:n { #2 } }
2357       } }
2358   }{
2359     \str_if_in:NnTF \l__stex_notation_args_str B {
2360       \exp_args:Nne \use:nn
2361       {
2362         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2363         \cs_set:Npn \l__stex_notation_arity_str } { {
2364           \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2365           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2366           { \l__stex_notation_opprec_tl }
2367           { \exp_not:n { #2 } }
2368         } }
2369     }{
2370       \exp_args:Nne \use:nn
2371       {
2372         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2373         \cs_set:Npn \l__stex_notation_arity_str } { {
2374           \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2375           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2376         { \l__stex_notation_opprec_tl }
2377         { \exp_not:n { #2 } }
2378     } }
2379 }
2380 }
2381
2382 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2383 \int_zero:N \l__stex_notation_currarg_int
2384 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2385 \__stex_notation_arguments:
2386 }
2387 }

```

(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

```

2388 \cs_new_protected:Nn \__stex_notation_arguments: {
2389   \int_incr:N \l__stex_notation_currarg_int
2390   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2391     \__stex_notation_final:
2392   }{
2393     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2394     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2395     \str_if_eq:NnTF \l_tmpa_str a {
2396       \__stex_notation_argument_assoc:n
2397     }{
2398       \str_if_eq:NnTF \l_tmpa_str B {
2399         \__stex_notation_argument_assoc:n
2400       }{
2401         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2402         \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2403           { \stex_term_math_arg:nnn
2404             { \int_use:N \l__stex_notation_currarg_int }
2405             { \l_tmpa_str }
2406             { ###\int_use:N \l__stex_notation_currarg_int }
2407           }
2408         }
2409         \__stex_notation_arguments:
2410       }
2411     }
2412   }
2413 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2414 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2415
2416   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2417     {\l__stex_notation_arity_str}{
2418     #1
2419   }
2420   \int_zero:N \l_tmpa_int
2421   \tl_clear:N \l_tmpa_tl

```



```

2472 }
2473 \cs_set:Npn \l__stex_notation_arity_str } { {
2474   \exp_after:wN \exp_after:wN \exp_after:wN
2475   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2476   { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2477 } }
2478
2479 \tl_if_empty:NF \l__stex_notation_op_tl {
2480   \cs_set:cpx {
2481     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2482     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2483     _cs
2484   } {
2485     \stex_term_oms:nnn {
2486       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2487       \l__stex_notation_lang_str
2488     }{
2489       \l__stex_notation_symbol_str
2490     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2491   }
2492 }
2493
2494 \exp_args:Ne
2495 \stex_add_to_current_module:n {
2496   \cs_generate_from_arg_count:cNnn {
2497     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2498     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2499     _cs
2500   } \cs_set:Npn {\l__stex_notation_arity_str} {
2501     \exp_after:wN \exp_after:wN \exp_after:wN
2502     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2503     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2504   }
2505   \tl_if_empty:NF \l__stex_notation_op_tl {
2506     \cs_set:cpn {
2507       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2508       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2509       _cs
2510     } {
2511       \stex_term_oms:nnn {
2512         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2513         \l__stex_notation_lang_str
2514       }{
2515         \l__stex_notation_symbol_str
2516       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2517     }
2518   }
2519 }
2520 \exp_args:Nx
2521 % \stex_do_aftergroup:n {
2522   \seq_put_right:cx {
2523     l_stex_symdecl_ \l__stex_notation_symbol_str
2524     _notations
2525   } {

```



```

2526     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2527 }
2528 % }
2529
2530 \stex_debug:nn{symbols}{
2531   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2532   ~for~\l__stex_notation_symbol_str^^J
2533   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2534   Argument~precedences:~
2535   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2536   Notation: \cs_meaning:c {
2537     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2538     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2539     _cs
2540   }
2541 }
2542
2543 %\prop_set_eq:cN {
2544 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2545 %   \c_hash_str \l__stex_notation_lang_str _prop
2546 %} \l_tmpb_prop
2547
2548 \exp_args:Ne
2549 \stex_add_to_current_module:n {
2550   \seq_put_right:cn {
2551     l_stex_symdecl_ \l__stex_notation_symbol_str
2552     _notations
2553   } {
2554     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2555   }
2556   %\prop_set_from_keyval:cn {
2557   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2558   %   \c_hash_str \l__stex_notation_lang_str _prop
2559   %} {
2560   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2561   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2562   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2563   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2564   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2565   %}
2566 }
2567
2568 \stex_if_smsmode:TF {
2569 %   \exp_args:Nx \stex_add_to_sms:n {
2570 %   \prop_set_from_keyval:cn {
2571 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2572 %   \c_hash_str \l__stex_notation_lang_str _prop
2573 %   } {
2574 %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2575 %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2576 %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2577 %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2578 %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2579 %   }

```

```

2580 %    }
2581 }{
2582
2583 % HTML annotations
2584 \stex_if_do_html:T {
2585   \stex_annotate_invisible:nnn { notation }
2586   { \l__stex_notation_symbol_str } {
2587     \stex_annotate_invisible:nnn { notationfragment }
2588     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2589     \stex_annotate_invisible:nnn { precedence }
2590     { \l__stex_notation_prec_str }{}
2591
2592     \int_zero:N \l_tmpa_int
2593     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2594     \tl_clear:N \l_tmpa_tl
2595     \int_step_inline:nn { \l__stex_notation_arity_str }{
2596       \int_incr:N \l_tmpa_int
2597       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2598       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2599       \str_if_eq:VnTF \l_tmpb_str a {
2600         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2601           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2602           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2603         } }
2604       }{
2605         \str_if_eq:VnTF \l_tmpb_str B {
2606           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2607             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2608             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2609           } }
2610         }{
2611           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2612             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2613           } }
2614         }
2615       }
2616     }
2617     \stex_annotate_invisible:nnn { notationcomp }{}{
2618       \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2619       $ \exp_args:Nno \use:nn { \use:c {
2620         stex_notation_ \l_stex_current_symbol_str
2621         \c_hash_str \l__stex_notation_variant_str
2622         \c_hash_str \l__stex_notation_lang_str _cs
2623       } } { \l_tmpa_tl } $
2624     }
2625   }
2626 }
2627 }
2628 \stex_smsmode_do:
2629 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

2630 \keys_define:nn { stex / setnotation } {
2631   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2632   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2633   unknown .code:n = \str_set:Nx
2634     \l__stex_notation_variant_str \l_keys_key_str
2635 }
2636
2637 \cs_new_protected:Nn \stex_setnotation_args:n {
2638   \str_clear:N \l__stex_notation_lang_str
2639   \str_clear:N \l__stex_notation_variant_str
2640   \keys_set:nn { stex / setnotation } { #1 }
2641 }
2642
2643 \NewDocumentCommand \setnotation {m m} {
2644   \stex_get_symbol:n { #1 }
2645   \stex_setnotation_args:n { #2 }
2646   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2647     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2648     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2649       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2650     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2651       { \c_hash_str }
2652     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2653       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2654     \exp_args:Nx \stex_add_to_current_module:n {
2655       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2656         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2657       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2658         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2659       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notati
2660         { \c_hash_str }
2661     }
2662     \stex_debug:nn {notations}{
2663       Setting~default~notation~
2664       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2665       \l_stex_get_symbol_uri_str \
2666       \expandafter\meaning\csname
2667         l_stex_symdecl \l_stex_get_symbol_uri_str _notations\endcsname
2668     }
2669   }{
2670     % todo throw error
2671   }
2672   \stex_smsmode_do:
2673 }
2674
2675 \cs_new_protected:Nn \stex_copy_notations:nn {
2676   \stex_debug:nn {notations}{
2677     Copying~notations~from~#2~to~#1\
2678     \seq_use:cn{l_stex_symdecl #2 _notations}{,~}
2679   }
2680   \tl_clear:N \l_tmpa_tl
2681   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl #2 _prop}{ arity } } {
2682     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2683 }

```

```

2684 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2685   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2686   \edef \l_tmpa_tl {
2687     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2688     \exp_after:wN\exp_after:wN\exp_after:wN {
2689       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2690     }
2691   }
2692   \exp_args:Nx
2693   \stex_do_aftergroup:n {
2694     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2695     \cs_generate_from_arg_count:cNnn {
2696       stex_notation_ #1 \c_hash_str ##1 _cs
2697     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2698       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2699     }
2700   }
2701 }
2702 }
2703
2704 \NewDocumentCommand \copynotation {m m} {
2705   \stex_get_symbol:n { #1 }
2706   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2707   \stex_get_symbol:n { #2 }
2708   \exp_args:Noo
2709   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2710   \exp_args:Nx \stex_add_import_to_current_module:n{
2711     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2712   }
2713   \stex_smsmode_do:
2714 }
2715

```

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

\symdef

```

2716 \keys_define:nn { stex / symdef } {
2717   name .str_set_x:N = \l_stex_symdecl_name_str ,
2718   local .bool_set:N = \l_stex_symdecl_local_bool ,
2719   args .str_set_x:N = \l_stex_symdecl_args_str ,
2720   type .tl_set:N = \l_stex_symdecl_type_tl ,
2721   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2722   op .tl_set:N = \l_stex_notation_op_tl ,
2723   lang .str_set_x:N = \l_stex_notation_lang_str ,
2724   variant .str_set_x:N = \l_stex_notation_variant_str ,
2725   prec .str_set_x:N = \l_stex_notation_prec_str ,
2726   assoc .choices:nn =
2727     {bin,binl,binr,pre,conj,pwconj}
2728     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2729   unknown .code:n = \str_set:Nx
2730     \l_stex_notation_variant_str \l_keys_key_str
2731 }
2732
2733 \cs_new_protected:Nn \_stex_notation_symdef_args:n {

```

```

2734 \str_clear:N \l_stex_symdecl_name_str
2735 \str_clear:N \l_stex_symdecl_args_str
2736 \str_clear:N \l_stex_symdecl_assoc_type_str
2737 \bool_set_false:N \l_stex_symdecl_local_bool
2738 \tl_clear:N \l_stex_symdecl_type_tl
2739 \tl_clear:N \l_stex_symdecl_definiens_tl
2740 \str_clear:N \l__stex_notation_lang_str
2741 \str_clear:N \l__stex_notation_variant_str
2742 \str_clear:N \l__stex_notation_prec_str
2743 \tl_clear:N \l__stex_notation_op_tl
2744
2745 \keys_set:nn { stex / symdef } { #1 }
2746 }
2747
2748 \NewDocumentCommand \symdef { O{} m } {
2749   \__stex_notation_symdef_args:n { #1 }
2750   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2751   \stex_symdecl_do:n { #2 }
2752   \exp_args:Nx \stex_notation_do:nn {
2753     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2754   }
2755 }
2756 \stex_deactivate_macro:Nn \symdef {module~environments}

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

2757 \endpackage

```

Chapter 31

STEX -Terms Implementation

```
2758 <*package>
2759
2760 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2761
2762 <@@=stex_terms>
2763
2764 Warnings and error messages
2765 \msg_new:nnn{stex}{error/nonotation}{
2766   Symbol~#1~invoked,~but~has~no~notation~#2!
2767 }
2768 \msg_new:nnn{stex}{error/notationarg}{
2769   Error~in~parsing~notation~#1
2770 }
2771 \msg_new:nnn{stex}{error/noop}{
2772   Symbol~#1~has~no~operator~notation~for~notation~#2
2773 }
```

31.1 Symbol Invocations

Arguments:

```
2773 \keys_define:nn { stex / terms } {
2774   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2775   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2776   unknown .code:n = \str_set:Nx
2777     \l__stex_terms_variant_str \l_keys_key_str
2778 }
2779
2780 \cs_new_protected:Nn \__stex_terms_args:n {
2781   \str_clear:N \l__stex_terms_lang_str
2782   \str_clear:N \l__stex_terms_variant_str
2783   \str_clear:N \l__stex_terms_prec_str
2784   \tl_clear:N \l__stex_terms_op_tl
2785 }
2786 \keys_set:nn { stex / terms } { #1 }
```

2787 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2788 \cs_new_protected:Nn \stex_invoke_symbol:n {
2789   \str_if_eq:eeF {
2790     \prop_item:cn {
2791       l_stex_symdecl_#1_prop
2792     }{ deprecate }
2793   }{}{
2794     \msg_warning:nxxx{stex}{warning/deprecated}{
2795       Symbol~#1
2796     }{
2797       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2798     }
2799   }
2800   \if_mode_math:
2801     \exp_after:wN \__stex_terms_invoke_math:n
2802   \else:
2803     \exp_after:wN \__stex_terms_invoke_text:n
2804   \fi: { #1 }
2805 }
```

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

`__stex_terms_invoke_math:n`

```

2806 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2807   \peek_charcode_remove:NTF ! {
2808     \peek_charcode:NTF [ {
2809       \__stex_terms_invoke_op:nw { #1 }
2810     }{
2811       \peek_charcode_remove:NTF ! {
2812         \peek_charcode:NTF [ {
2813           \__stex_terms_invoke_op_custom:nw
2814         }{
2815           % TODO throw error
2816         }
2817       }{
2818         \__stex_terms_invoke_op:nw { #1 } []
2819       }
2820     }
2821   }{
2822     \peek_charcode_remove:NTF * {
2823       \__stex_terms_invoke_text:n { #1 }
2824     }{
2825       \peek_charcode:NTF [ {
2826         \__stex_terms_invoke_math:nw { #1 }
2827       }{
2828         \__stex_terms_invoke_math:nw { #1 } []
2829       }
2830     }
2831   }
2832 }
```

(End definition for `__stex_terms_invoke_math:n`.)

_stex_terms_invoke_op_custom:nw

```

2833 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2834   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2835     \stex_highlight_term:nn{#1}{#2}
2836   }
2837 }

```

(End definition for _stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2838 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2839   \_stex_terms_args:n { #2 }
2840   \cs_if_exist:cTF {
2841     stex_op_notation_ #1 \c_hash_str
2842     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2843   }{
2844     \csname stex_op_notation_ #1 \c_hash_str
2845       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2846     \endcsname
2847   }{
2848     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2849   }
2850 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2851 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2852   \_stex_terms_args:n { #2 }
2853   \seq_if_empty:cTF {
2854     l_stex_symdecl_ #1 _notations
2855   } {
2856     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2857   } {
2858     \seq_if_in:cxTF {
2859       l_stex_symdecl_ #1 _notations
2860     }
2861     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2862       \str_set:Nn \l_stex_current_symbol_str { #1 }
2863       \stex_debug:nn{terms}{Using~
2864         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2865       \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2866       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2867       _cs\endcsname
2868     }
2869     \use:c{
2870       stex_notation_ #1 \c_hash_str
2871       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2872       _cs
2873     }
2874   }{
2875     \str_if_empty:NTF \l__stex_terms_variant_str {
2876       \str_if_empty:NTF \l__stex_terms_lang_str {
2877         \seq_get_left:cN {

```



```

2878         l_stex_symdecl_ #1 _notations
2879     } \l_tmpa_str
2880     \str_set:Nn \l_stex_current_symbol_str { #1 }
2881     \stex_debug:nn{terms}{Using~
2882         #1\c_hash_str\l_tmpa_str \\
2883         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2884         \l_tmpa_str
2885         _cs\endcsname
2886     }
2887     \use:c{
2888         stex_notation_ #1 \c_hash_str \l_tmpa_str
2889         _cs
2890     }
2891 }{
2892     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2893         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2894     }
2895 }
2896 }{
2897     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2898         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2899     }
2900 }
2901 }
2902 }
2903 }

```

(End definition for _stex_terms_invoke_math:nw.)

_stex_terms_invoke_text:n

```

2904 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2905     \peek_charcode_remove:NTF ! {
2906         \stex_term_custom:nn { #1 } { }
2907     }{
2908         \prop_set_eq:Nc \l_tmpa_prop {
2909             l_stex_symdecl_ #1 _prop
2910         }
2911         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2912         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2913     }
2914 }

```

(End definition for _stex_terms_invoke_text:n.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2915 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2916 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2917 \int_new:N \l__stex_terms_downprec
2918 \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`

```
2919 \tl_set:Nn \l__stex_terms_left_bracket_str (
2920 \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

```
2921 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2922   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2923     \bool_set_false:N \l__stex_terms_brackets_done_bool
2924     #2
2925   } {
2926     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2927       \bool_if:NTF \l__stex_inarray_bool { #2 } {
2928         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2929         \dobrackets { #2 }
2930       }
2931     }{ #2 }
2932   }
2933 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2934 \bool_new:N \l__stex_terms_brackets_done_bool
2935 %\RequirePackage{scalerel}
2936 \cs_new_protected:Npn \dobrackets #1 {
2937   %\ThisStyle{\if D\m@switch
2938   %   \exp_args:Nnx \use:nn
2939   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2940   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2941   %   \else
2942   %   \exp_args:Nnx \use:nn
2943   %   {
2944     \bool_set_true:N \l__stex_terms_brackets_done_bool
2945     \int_set:Nn \l__stex_terms_downprec \infprec
2946     \l__stex_terms_left_bracket_str
2947     #1
2948   }
2949   {
2950     \bool_set_false:N \l__stex_terms_brackets_done_bool
2951     \l__stex_terms_right_bracket_str
2952     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2953   }
2954   %\fi}
2955 }
```

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

\withbrackets

```
2956 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2957   \exp_args:Nnx \use:nn
2958   {
2959     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2960     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2961     #3
2962   }
2963   {
2964     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2965     {\l__stex_terms_left_bracket_str}
2966     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2967     {\l__stex_terms_right_bracket_str}
2968   }
2969 }
```

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

\STEXinvisible

```
2970 \cs_new_protected:Npn \STEXinvisible #1 {
2971   \stex_annotate_invisible:n { #1 }
2972 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
2973 \cs_new_protected:Nn \_stex_term_oms:nnn {
2974   \stex_annotate:nnn{ OMID }{ #2 }{
2975     \stex_highlight_term:nn { #1 } { #3 }
2976   }
2977 }
2978
2979 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2980   \_stex_terms_maybe_brackets:nn { #3 }{
2981     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2982   }
2983 }
```

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

_stex_term_math_oma:nnnn

```
2984 \cs_new_protected:Nn \_stex_term_oma:nnn {
2985   \stex_annotate:nnn{ OMA }{ #2 }{
2986     \stex_highlight_term:nn { #1 } { #3 }
2987   }
2988 }
2989
2990 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2991   \_stex_terms_maybe_brackets:nn { #3 }{
2992     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2993   }
2994 }
```

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

`_stex_term_math_omb:nnnn`

```

2995 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2996   \stex_annotate:nnn{ OMBIND }{ #2 }{
2997     \stex_highlight_term:nn { #1 } { #3 }
2998   }
2999 }
3000
3001 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3002   \__stex_terms_maybe_brackets:nn { #3 }{
3003     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3004   }
3005 }

```

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

`_stex_term_math_arg:nnn`

```

3006 \cs_new_protected:Nn \_stex_term_arg:nn {
3007   \stex_unhighlight_term:n {
3008     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3009   }
3010 }
3011 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3012   \exp_args:Nnx \use:nn
3013   { \int_set:Nn \l__stex_terms_downprec { #2 }
3014     \stex_term_arg:nn { #1 }{ #3 }
3015   }
3016   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3017 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3018 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3019   % TODO sequences
3020   \clist_set:Nn \l_tmpa_clist{ #3 }
3021   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3022     \tl_set:Nn \l_tmpa_tl { #3 }
3023   }{
3024     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3025     \clist_reverse:N \l_tmpa_clist
3026     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3027
3028     \clist_map_inline:Nn \l_tmpa_clist {
3029       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3030         \exp_args:Nno
3031         \l_tmpa_cs { ##1 } \l_tmpa_tl
3032       }
3033     }
3034   }
3035   \exp_args:Nnno
3036   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3037 }

```

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

`\stex_term_custom:nn`

```

3038 \cs_new_protected:Nn \stex_term_custom:nn {
3039   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3040   \str_set:Nn \l_tmpa_str { #2 }
3041   \tl_clear:N \l_tmpa_tl
3042   \int_zero:N \l_tmpa_int
3043   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3044   \__stex_terms_custom_loop:
3045 }

```

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

`__stex_terms_custom_loop:`

```

3046 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3047   \bool_set_false:N \l_tmpa_bool
3048   \bool_while_do:nn {
3049     \str_if_eq_p:ee X {
3050       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3051     }
3052   }{
3053     \int_incr:N \l_tmpa_int
3054   }
3055
3056   \peek_charcode:NTF [ {
3057     % notation/text component
3058     \__stex_terms_custom_component:w
3059   } {
3060     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3061       % all arguments read => finish
3062       \__stex_terms_custom_final:
3063     } {
3064       % arguments missing
3065       \peek_charcode_remove:NTF * {
3066         % invisible, specific argument position or both
3067         \peek_charcode:NTF [ {
3068           % visible specific argument position
3069           \__stex_terms_custom_arg:wn
3070         } {
3071           % invisible
3072           \peek_charcode_remove:NTF * {
3073             % invisible specific argument position
3074             \__stex_terms_custom_arg_inv:wn
3075           } {
3076             % invisible next argument
3077             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3078           }
3079         }
3080       } {
3081         % next normal argument
3082         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3083       }
3084     }
3085   }
3086 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

3087 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3088   \bool_set_true:N \l_tmpa_bool
3089   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3090 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

3091 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3092   \str_set:Nx \l_tmpb_str {
3093     \str_item:Nn \l_tmpa_str { #1 }
3094   }
3095   \str_case:VnTF \l_tmpb_str {
3096     { X } {
3097       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3098     }
3099     { i } { \_stex_terms_custom_set_X:n { #1 } }
3100     { b } { \_stex_terms_custom_set_X:n { #1 } }
3101     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3102     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3103   }{}{
3104     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3105   }
3106
3107   \bool_if:nTF \l_tmpa_bool {
3108     \tl_put_right:Nx \l_tmpa_tl {
3109       \stex_annotate_invisible:n {
3110         \stex_term_arg:nn { \int_eval:n { #1 } }
3111         \exp_not:n { { #2 } }
3112       }
3113     }
3114   } {
3115     \tl_put_right:Nx \l_tmpa_tl {
3116       \stex_term_arg:nn { \int_eval:n { #1 } }
3117       \exp_not:n { { #2 } }
3118     }
3119   }
3120
3121   \_stex_terms_custom_loop:
3122 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

3123 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3124   \str_set:Nx \l_tmpa_str {
3125     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3126     X
3127     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3128   }
3129 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

3130 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3131   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3132   \_stex_terms_custom_loop:
3133 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

3134 \cs_new_protected:Nn \_stex_terms_custom_final: {
3135   \int_compare:nNnTF \l_tmpb_int = 0 {
3136     \exp_args:Nnno \_stex_term_oms:nnn
3137   }{
3138     \str_if_in:NnTF \l_tmpa_str {b} {
3139       \exp_args:Nnno \_stex_term_ombind:nnn
3140     } {
3141       \exp_args:Nnno \_stex_term_oma:nnn
3142     }
3143   }
3144   { \l_stex_terms_custom_uri } { \l_stex_terms_custom_uri } { \l_tmpa_tl }
3145 }

```

(End definition for _stex_terms_custom_final:.)

\symref
\symname

```

3146 \NewDocumentCommand \symref { m m }{
3147   \let\compemph_uri_prev:\compemph@uri
3148   \let\compemph@uri\symrefemph@uri
3149   \STEXsymbol{#1}!{#2}
3150   \let\compemph@uri\compemph_uri_prev:
3151 }
3152
3153 \keys_define:nn { stex / symname } {
3154   post .str_set_x:N = \l_stex_symname_post_str
3155 }
3156
3157 \cs_new_protected:Nn \stex_symname_args:n {
3158   \str_clear:N \l_stex_symname_post_str
3159   \keys_set:nn { stex / symname } { #1 }
3160 }
3161
3162 \NewDocumentCommand \symname { 0{ } m }{
3163   \stex_symname_args:n { #1 }
3164   \stex_get_symbol:n { #2 }
3165   \str_set:Nx \l_tmpa_str {
3166     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3167   }
3168   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3169
3170   \let\compemph_uri_prev:\compemph@uri
3171   \let\compemph@uri\symrefemph@uri
3172   \exp_args:NNx \use:nn

```

```

3173 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3174 \l_tmpa_str \l_stex_symname_post_str
3175 ] }
3176 \let\compemph@uri\compemph_uri_prev:
3177 }

```

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

31.3 Notation Components

```

3178 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3179
3180 \str_new:N \l_stex_current_symbol_str
3181 \cs_new_protected:Nn \stex_highlight_term:nn {
3182   \exp_args:Nnx
3183   \use:nn {
3184     \str_set:Nx \l_stex_current_symbol_str { #1 }
3185     #2
3186   } {
3187     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3188     { \l_stex_current_symbol_str }
3189   }
3190 }
3191
3192 \cs_new_protected:Nn \stex_unhighlight_term:n {
3193   % \latexml_if:TF {
3194   %   #1
3195   % } {
3196   %   \rustex_if:TF {
3197   %     #1
3198   %   } {
3199     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3200   % }
3201   % }
3202 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3203 \cs_new_protected:Npn \comp #1 {
3204   \str_if_empty:NF \l_stex_current_symbol_str {
3205     \rustex_if:TF {
3206       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3207     }{
3208       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3209     }
3210   }
3211 }
3212
3213 \cs_new_protected:Npn \compemph@uri #1 #2 {
3214   \compemph{ #1 }
3215 }

```



```

3216
3217
3218 \cs_new_protected:Npn \compemph #1 {
3219     #1
3220 }
3221
3222 \cs_new_protected:Npn \defemph@uri #1 #2 {
3223     \defemph{#1}
3224 }
3225
3226 \cs_new_protected:Npn \defemph #1 {
3227     \textbf{#1}
3228 }
3229
3230 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3231     \symrefemph{#1}
3232 }
3233
3234 \cs_new_protected:Npn \symrefemph #1 {
3235     \textbf{#1}
3236 }

```

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

\ellipses

```

3237 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3238 \bool_new:N \l_stex_inarray_bool
3239 \bool_set_false:N \l_stex_inarray_bool
3240 \NewDocumentCommand \parray { m m } {
3241     \begin{group}
3242     \bool_set_true:N \l_stex_inarray_bool
3243     \begin{array}{#1}
3244         #2
3245     \end{array}
3246     \end{group}
3247 }
3248
3249 \NewDocumentCommand \prmatrix { m } {
3250     \begin{group}
3251     \bool_set_true:N \l_stex_inarray_bool
3252     \begin{matrix}
3253         #1
3254     \end{matrix}
3255     \end{group}
3256 }
3257
3258 \def \maybepline {
3259     \bool_if:NT \l_stex_inarray_bool {\hline}
3260 }
3261
3262 \def \parrayline #1 #2 {

```

```

3263   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3264 }
3265
3266 \def \pmrow #1 { \parrayline{ }{ #1 } }
3267
3268 \def \parraylineh #1 #2 {
3269   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3270 }
3271
3272 \def \parraycell #1 {
3273   #1 \bool_if:NT \l_stex_inarray_bool {&}
3274 }

```

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

```

3275 \endpackage

```

Chapter 32

STEX -Structural Features Implementation

```
3276 <*package>
3277
3278 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3279
3280 <@@=stex_features>
3281
3282   Warnings and error messages
3283 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3284   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3285 }
3286 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3287   Symbol~#1~not~assigned~in~interpretmodule~#2
3288 }
3289
```

32.1 Imports with modification

```
3288 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3289   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3290     \__stex_features_get_symbol_from_cs:n { #1 }
3291   }{
3292     % argument is a string
3293     % is it a command name?
3294     \cs_if_exist:cTF { #1 }{
3295       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3296       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3297       \str_if_empty:NNTF \l_tmpa_str {
3298         \exp_args:Nx \cs_if_eq:NNTF {
3299           \tl_head:N \l_tmpa_tl
3300         } \stex_invoke_symbol:n {
3301           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3302         }{
3303           \__stex_features_get_symbol_from_string:n { #1 }
3304         }
3305       }
3306     }
3307   }
3308 }
```

```

3304     }
3305   } {
3306     \__stex_features_get_symbol_from_string:n { #1 }
3307   }
3308   ){
3309     % argument is not a command name
3310     \__stex_features_get_symbol_from_string:n { #1 }
3311     % \l_stex_all_symbols_seq
3312   }
3313 }
3314 }
3315
3316 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3317   \str_set:Nn \l_tmpa_str { #1 }
3318   \bool_set_false:N \l_tmpa_bool
3319   \bool_if:NF \l_tmpa_bool {
3320     \tl_set:Nn \l_tmpa_tl {
3321       \msg_set:nnn{stex}{error/unknownsymbol}{
3322         No~symbol~#1~found!
3323       }
3324       \msg_error:nn{stex}{error/unknownsymbol}
3325     }
3326     \str_set:Nn \l_tmpa_str { #1 }
3327     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3328     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3329       \str_set:Nn \l_tmpb_str { ##1 }
3330       \str_if_eq:eeT { \l_tmpa_str } {
3331         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3332       } {
3333         \seq_map_break:n {
3334           \tl_set:Nn \l_tmpa_tl {
3335             \str_set:Nn \l_stex_get_symbol_uri_str {
3336               ##1
3337             }
3338             \__stex_features_get_symbol_check:
3339           }
3340         }
3341       }
3342     }
3343     \l_tmpa_tl
3344   }
3345 }
3346
3347 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3348   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3349   { \tl_tail:N \l_tmpa_tl }
3350   \tl_if_single:NTF \l_tmpa_tl {
3351     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3352       \exp_after:wN \str_set:Nn \exp_after:wN
3353       \l_stex_get_symbol_uri_str \l_tmpa_tl
3354       \__stex_features_get_symbol_check:
3355     }{
3356       % TODO
3357       % tail is not a single group

```

```

3358     }
3359   }{
3360     % TODO
3361     % tail is not a single group
3362   }
3363 }
3364
3365 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3366   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3367   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3368     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3369     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3370     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3371       \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3372         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3373       }
3374     }
3375   }{
3376     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3377       \l_stex_current_copymodule_name_str~(inexplicably)
3378     }
3379   }
3380 }
3381
3382 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3383   \stex_import_module_uri:nn { #1 } { #2 }
3384   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3385   \stex_import_require_module:nnnn
3386     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3387     { \l_stex_import_path_str } { \l_stex_import_name_str }
3388   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3389   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3390   \seq_clear:N \l__stex_features_copymodule_fields_seq
3391   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3392     \seq_map_inline:cn {c_stex_module_###1_constants}{
3393       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3394         ###1 ? ####1
3395       }
3396     }
3397   }
3398   \seq_clear:N \l_tmpa_seq
3399   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3400     name      = \l_stex_current_copymodule_name_str ,
3401     module    = \l_stex_current_module_str ,
3402     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3403     includes  = \l_tmpa_seq ,
3404     fields    = \l_tmpa_seq
3405   }
3406   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3407     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3408   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3409     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3410   \stex_if_smsmode:F {
3411     \begin{stex_annotate_env} {#4} {

```

```

3412     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3413   }
3414   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3415 }
3416 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3417 \bool_set_false:N \l_stex_html_do_output_bool
3418 }
3419 \cs_new_protected:Nn \stex_copymodule_end:n {
3420   \def \l_tmpa_cs ##1 ##2 {#1}
3421   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3422   \tl_clear:N \l_tmpa_tl
3423   \tl_clear:N \l_tmpb_tl
3424   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3425   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3426     \seq_map_inline:cn {c_stex_module_##1_constants}{
3427       \tl_clear:N \l_tmpc_tl
3428       \l_tmpa_cs{##1}{####1}
3429       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3430         \tl_put_right:Nx \l_tmpa_tl {
3431           \prop_set_from_keyval:cn {
3432             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3433           }{
3434             \exp_after:wN \prop_to_keyval:N \csname
3435               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3436             \endcsname
3437           }
3438           \seq_clear:c {
3439             l_stex_symdecl_
3440             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3441             _notations
3442           }
3443         }
3444         \tl_put_right:Nx \l_tmpc_tl {
3445           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3446           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3447         }
3448         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3449         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3450           \tl_put_right:Nx \l_tmpc_tl {
3451             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3452           }
3453           \tl_put_right:Nx \l_tmpa_tl {
3454             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\}
3455             \stex_invoke_symbol:n {
3456               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3457             }
3458           }
3459         }
3460       }
3461     }{
3462       \tl_put_right:Nx \l_tmpc_tl {
3463         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3464       }
3465       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3466 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3467 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3468 \tl_put_right:Nx \l_tmpa_tl {
3469   \prop_set_from_keyval:cn {
3470     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3471   }{
3472     \prop_to_keyval:N \l_tmpa_prop
3473   }
3474   \seq_clear:c {
3475     l_stex_symdecl_
3476     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3477     _notations
3478   }
3479 }
3480 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3481 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3482   \tl_put_right:Nx \l_tmpc_tl {
3483     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3484   }
3485   \tl_put_right:Nx \l_tmpa_tl {
3486     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3487       \stex_invoke_symbol:n {
3488         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3489       }
3490     }
3491   }
3492 }
3493 }
3494 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3495   \tl_put_right:Nx \l_tmpc_tl {
3496     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3497   }
3498 }
3499 \tl_put_right:Nx \l_tmpb_tl {
3500   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3501 }
3502 }
3503 }
3504 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3505 \tl_put_left:Nx \l_tmpa_tl {
3506   \prop_set_from_keyval:cn {
3507     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3508   }{
3509     \prop_to_keyval:N \l_stex_current_copymodule_prop
3510   }
3511 }
3512 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3513 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3514 \exp_args:Nx \stex_do_aftergroup:n {
3515   \exp_args:No \exp_not:n \l_tmpa_tl
3516 }
3517 \l_tmpb_tl
3518 \stex_if_smsmode:F {
3519   \end{stex_annotate_env}

```

```

3520 }
3521 }
3522
3523 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3524   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3525   \stex_deactivate_macro:Nn \symdecl {module~environments}
3526   \stex_deactivate_macro:Nn \symdef {module~environments}
3527   \stex_deactivate_macro:Nn \notation {module~environments}
3528   \stex_reactivate_macro:N \assign
3529   \stex_reactivate_macro:N \renamedec1
3530   \stex_reactivate_macro:N \donotcopy
3531   \stex_smsmode_do:
3532 }{
3533   \stex_copymodule_end:n {}
3534 }
3535
3536 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3537   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3538   \stex_deactivate_macro:Nn \symdecl {module~environments}
3539   \stex_deactivate_macro:Nn \symdef {module~environments}
3540   \stex_deactivate_macro:Nn \notation {module~environments}
3541   \stex_reactivate_macro:N \assign
3542   \stex_reactivate_macro:N \renamedec1
3543   \stex_reactivate_macro:N \donotcopy
3544   \stex_smsmode_do:
3545 }{
3546   \stex_copymodule_end:n {
3547     \tl_if_exist:cF {
3548       l__stex_features_copymodule_##1?##2_def_tl
3549     }{
3550       \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
3551         ##1?##2
3552       }{\l_stex_current_copymodule_name_str}
3553     }
3554   }
3555 }
3556
3557 \NewDocumentCommand \donotcopy { 0{} m}{
3558   \stex_import_module_uri:nn { #1 } { #2 }
3559   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3560   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3561     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3562     \seq_map_inline:cn {c_stex_module_##1_constants}{
3563       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3564       \bool_lazy_any_p:nT {
3565         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3566         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3567         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3568       }{
3569         % TODO throw error
3570       }
3571     }
3572   }
3573 }

```



```

3574 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3575 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3576 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3577 }
3578
3579 \NewDocumentCommand \assign { m m }{
3580 \stex_get_symbol_in_copymodule:n {#1}
3581 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3582 \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3583 }
3584
3585 \keys_define:nn { stex / renamedec1 } {
3586   name .str_set_x:N = \l_stex_renamedec1_name_str
3587 }
3588 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3589   \str_clear:N \l_stex_renamedec1_name_str
3590
3591   \keys_set:nn { stex / renamedec1 } { #1 }
3592 }
3593
3594 \NewDocumentCommand \renamedec1 { 0{} m m }{
3595   \__stex_features_renamedec1_args:n { #1 }
3596   \stex_get_symbol_in_copymodule:n {#2}
3597   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3598   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3599   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3600     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3601       \l_stex_get_symbol_uri_str
3602     } }
3603   } {
3604     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3605     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3606     \prop_set_eq:cc {l_stex_symdecl_
3607       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3608     _prop
3609     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3610     \seq_set_eq:cc {l_stex_symdecl_
3611       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3612     _notations
3613     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3614     \prop_put:cnx {l_stex_symdecl_
3615       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3616     _prop
3617     }{ name }{ \l_stex_renamedec1_name_str }
3618     \prop_put:cnx {l_stex_symdecl_
3619       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3620     _prop
3621     }{ module }{ \l_stex_current_module_str }
3622     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3623       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3624     }
3625     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3626       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3627     } }

```

```

3628 }
3629 }
3630 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3631 % \_stex_notation_args:n { #1 }
3632 % \tl_clear:N \l_stex_symdecl_definiens_tl
3633 % \stex_get_symbol_in_copymodule:n { #2 }
3634 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3635 % % todo
3636 %}
3637 \stex_deactivate_macro:Nn \assign {copymodules}
3638 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3639 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3640
3641
3642 \seq_new:N \l_stex_implicit_morphisms_seq
3643 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3644 \stex_import_module_uri:nn { #1 } { #2 }
3645 \stex_debug:nn{implicits}{
3646 Implicit~morphism:~
3647 \l_stex_module_ns_str ? \l__stex_features_name_str
3648 }
3649 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3650 \l_stex_module_ns_str ? \l__stex_features_name_str
3651 }{
3652 \msg_error:nnn{stex}{error/conflictingmodules}{
3653 \l_stex_module_ns_str ? \l__stex_features_name_str
3654 }
3655 }
3656
3657 % TODO
3658
3659
3660
3661 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3662 \l_stex_module_ns_str ? \l__stex_features_name_str
3663 }
3664 }
3665

```

32.2 The feature environment

structural@feature

```

3666
3667 \NewDocumentEnvironment{structural@feature}{ m m m }{
3668 \stex_if_in_module:F {
3669 \msg_set:nnn{stex}{error/nomodule}{
3670 Structural~Feature~has~to~occur~in~a~module:\\
3671 Feature~#2~of~type~#1\\
3672 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3673 }
3674 \msg_error:nn{stex}{error/nomodule}
3675 }
3676

```

```

3677 \str_set:Nx \l_stex_module_name_str {
3678   \prop_item:Nn \l_stex_current_module_prop
3679     { name } / #2 - feature
3680 }
3681
3682 \str_set:Nx \l_stex_module_ns_str {
3683   \prop_item:Nn \l_stex_current_module_prop
3684     { ns }
3685 }
3686
3687
3688 \str_clear:N \l_tmpa_str
3689 \seq_clear:N \l_tmpa_seq
3690 \tl_clear:N \l_tmpa_tl
3691 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3692   origname = #2,
3693   name      = \l_stex_module_name_str ,
3694   ns        = \l_stex_module_ns_str ,
3695   imports   = \exp_not:o { \l_tmpa_seq } ,
3696   constants = \exp_not:o { \l_tmpa_seq } ,
3697   content   = \exp_not:o { \l_tmpa_tl } ,
3698   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3699   lang      = \l_stex_module_lang_str ,
3700   sig       = \l_tmpa_str ,
3701   meta      = \l_tmpa_str ,
3702   feature   = #1 ,
3703 }
3704
3705 \stex_if_smsmode:F {
3706   \begin{stex_annotate_env}{ feature:#1 }{}
3707   \stex_annotate_invisible:nnn{header}{}{ #3 }
3708 }
3709 }{
3710   \str_set:Nx \l_tmpa_str {
3711     c_stex_feature_
3712     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3713     \prop_item:Nn \l_stex_current_module_prop { name }
3714     _prop
3715   }
3716   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3717   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3718   \stex_if_smsmode:TF {
3719     \exp_args:Nx \stex_add_to_sms:n {
3720       \prop_gset_from_keyval:cn {
3721         c_stex_feature_
3722         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3723         \prop_item:Nn \l_stex_current_module_prop { name }
3724         _prop
3725       } {
3726         origname = #2,
3727         name      = \prop_item:cn { \l_tmpa_str } { name } ,
3728         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
3729         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
3730         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3731         content = \prop_item:cn { \l_tmpa_str } { content } ,
3732         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3733         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3734         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3735         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3736         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3737     }
3738 }
3739 } {
3740     \end{stex_annotate_env}
3741 }
3742 }
3743

```

32.3 Features

structure

```

3744
3745 \prop_new:N \l_stex_all_structures_prop
3746
3747 \keys_define:nn { stex / features / structure } {
3748     name .str_set_x:N = \l__stex_features_structure_name_str ,
3749 }
3750
3751 \cs_new_protected:Nn \__stex_features_structure_args:n {
3752     \str_clear:N \l__stex_features_structure_name_str
3753     \keys_set:nn { stex / features / structure } { #1 }
3754 }
3755
3756 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3757 % \__stex_features_structure_args:n { ##1 }
3758 % \str_if_empty:NT \l__stex_features_structure_name_str {
3759 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3760 % }
3761 %} {
3762 %
3763 %}
3764
3765 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3766     \__stex_features_structure_args:n { #1 }
3767     \str_if_empty:NT \l__stex_features_structure_name_str {
3768         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3769     }
3770     \exp_args:Nnnx
3771     \begin{structural@feature}{ structure }
3772         { \l__stex_features_structure_name_str }{}
3773         \seq_clear:N \l_tmpa_seq
3774         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3775     \stex_smsmode_do:
3776 }{
3777     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3778     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3779     \str_set:Nx \l_tmpa_str {

```

```

3780     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3781     \prop_item:Nn \l_stex_current_module_prop { name }
3782   }
3783   \seq_map_inline:Nn \l_tmpa_seq {
3784     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3785   }
3786   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3787   \exp_args:Nnx
3788   \AddToHookNext { env / mathstructure / after }{
3789     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3790       \_stex_term_math_oms:nxxx { \l_tmpa_str }{}{}{}{}
3791     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3792     \STEXexport {
3793       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3794         {\prop_item:Nn \l_stex_current_module_prop { origname }}
3795         {\l_tmpa_str}
3796       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3797         {#2}{\l_tmpa_str}
3798     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3799     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3800     %     \l_tmpa_str
3801     %   }
3802     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3803     %     #2,\l_tmpa_str
3804     %   }
3805     %   \tl_set:cx { #2 } {
3806     %     \stex_invoke_structure:n { \l_tmpa_str }
3807   }
3808 }
3809
3810 \end{structural@feature}
3811 % \g_stex_last_feature_prop
3812 }

```

\instantiate

```

3813 \seq_new:N \l__stex_features_structure_field_seq
3814 \str_new:N \l__stex_features_structure_field_str
3815 \str_new:N \l__stex_features_structure_def_tl
3816 \prop_new:N \l__stex_features_structure_prop
3817 \NewDocumentCommand \instantiate { m O{} m }{
3818   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3819   \prop_set_eq:Nc \l__stex_features_structure_prop {
3820     c_stex_feature_\l_tmpa_str _prop
3821   }
3822   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3823   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3824     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3825     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3826       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3827       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3828         {!} \l_tmpa_tl
3829       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3830         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3831         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3832     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3833   }{
3834     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3835     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3836     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3837       \l_tmpa_tl
3838     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3839       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3840       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3841     }{
3842       \tl_clear:N \l_tmpb_tl
3843     }
3844   }
3845 }{
3846   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3847   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3848     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3849     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3850     \tl_clear:N \l_tmpa_tl
3851   }{
3852     % TODO throw error
3853   }
3854 }
3855 % \l_tmpa_str: name
3856 % \l_tmpa_tl: definiens
3857 % \l_tmpb_tl: notation
3858 \tl_if_empty:NT \l__stex_features_structure_field_str {
3859   % TODO throw error
3860 }
3861 \str_clear:N \l_tmpb_str
3862
3863 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3864 \seq_map_inline:Nn \l_tmpa_seq {
3865   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3866   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3867   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3868     \seq_map_break:n {
3869       \str_set:Nn \l_tmpb_str { ####1 }
3870     }
3871   }
3872 }
3873 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3874   \l_tmpb_str
3875
3876 \tl_if_empty:NTF \l_tmpb_tl {
3877   \tl_if_empty:NF \l_tmpa_tl {
3878     \exp_args:Nx \use:n {
3879       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3880     }
3881   }
3882 }{
3883   \tl_if_empty:NTF \l_tmpa_tl {
3884     \exp_args:Nx \use:n {
3885       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3886     }
3887
3888   }{
3889     \exp_args:Nx \use:n {
3890       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3891       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3892     }
3893   }
3894 }
3895 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3896 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3897 % #3/\l__stex_features_structure_field_str
3898 % \par
3899 % \expandafter\present\csname
3900 %   \l_stex_symdecl_
3901 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3902 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3903 %   #3/\l__stex_features_structure_field_str
3904 %   _prop
3905 % \endcsname
3906 }
3907
3908 \tl_clear:N \l__stex_features_structure_def_tl
3909
3910 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3911 \seq_map_inline:Nn \l_tmpa_seq {
3912   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3913   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3914   \exp_args:Nx \use:n {
3915     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3916
3917     }
3918   }
3919
3920   \prop_if_exist:cF {
3921     \l_stex_symdecl_
3922     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3923     \prop_item:Nn \l_stex_current_module_prop {name} ?
3924     #3/\l_tmpa_str
3925     _prop
3926   }{
3927     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3928     \l_tmpb_str
3929     \exp_args:Nx \use:n {
3930       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3931     }
3932   }
3933 }
3934
3935 \symdecl*[type={\STEXsymbol{module-type}}{
3936   \_stex_term_math_oms:nnnn {
3937     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3938     \prop_item:Nn \l__stex_features_structure_prop {name}
3939   }-{}{0}-{}

```

```

3940   }}]{#3}
3941
3942   % TODO: -> sms file
3943
3944   \tl_set:cx{ #3 }{
3945     \stex_invoke_structure:nnn {
3946       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3947       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3948     } {
3949       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3950       \prop_item:Nn \l__stex_features_structure_prop {name}
3951     }
3952   }
3953   \stex_smsmode_do:
3954 }

```

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

`\stex_invoke_structure:nnn`

```

3955 % #1: URI of the instance
3956 % #2: URI of the instantiated module
3957 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3958   \tl_if_empty:nTF{ #3 }{
3959     \prop_set_eq:Nc \l__stex_features_structure_prop {
3960       c_stex_feature_ #2 _prop
3961     }
3962     \tl_clear:N \l_tmpa_tl
3963     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3964     \seq_map_inline:Nn \l_tmpa_seq {
3965       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3966       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3967       \cs_if_exist:cT {
3968         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3969       }{
3970         \tl_if_empty:NF \l_tmpa_tl {
3971           \tl_put_right:Nn \l_tmpa_tl {,}
3972         }
3973         \tl_put_right:Nx \l_tmpa_tl {
3974           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3975         }
3976       }
3977     }
3978     \exp_args:No \mathstrut \l_tmpa_tl
3979   }{
3980     \stex_invoke_symbol:n{#1/#3}
3981   }
3982 }

```

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

3983 `\</package>`

Chapter 33

STEX -Statements Implementation

```
3984 <*package>
3985
3986 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3987
3988 <@@=stex_statements>
3989
3990 \titleemph
3991 \def\titleemph#1{\textbf{#1}}
3992
3993 (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
3991 \keys_define:nn {stex / definiendum }{
3992   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3993   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3994   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3995 }
3996 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3997   \str_clear:N \l__stex_statements_definiendum_root_str
3998   \tl_clear:N \l__stex_statements_definiendum_post_tl
3999   \str_clear:N \l__stex_statements_definiendum_gfa_str
4000   \keys_set:nn { stex / definiendum }{ #1 }
4001 }
4002 \NewDocumentCommand \definiendum { 0{} m m } {
4003   \__stex_statements_definiendum_args:n { #1 }
4004   \stex_get_symbol:n { #2 }
4005   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
4006   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4007     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
4008       \tl_set:Nn \l_tmpa_tl { #3 }
4009     }
4010   }
4011 }
```

```

4009   } {
4010     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4011     \tl_set:Nn \l_tmpa_tl {
4012       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
4013     }
4014   }
4015 } {
4016   \tl_set:Nn \l_tmpa_tl { #3 }
4017 }
4018
4019 % TODO root
4020 \rustex_if:TF {
4021   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4022 } {
4023   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4024 }
4025 }
4026 \stex_deactivate_macro:Nn \definiendum {definition-environments}

```

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

definame

```

4027
4028 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4029
4030 \NewDocumentCommand \definame { 0{ } m } {
4031   \__stex_statements_definiendum_args:n { #1 }
4032   % TODO: root
4033   \stex_get_symbol:n { #2 }
4034   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4035   \str_set:Nx \l_tmpa_str {
4036     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4037   }
4038   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4039   \rustex_if:TF {
4040     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4041       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4042     }
4043   } {
4044     \defemph@uri {
4045       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4046     } { \l_stex_get_symbol_uri_str }
4047   }
4048 }
4049 \stex_deactivate_macro:Nn \definame {definition-environments}
4050
4051 \NewDocumentCommand \Definame { 0{ } m } {
4052   \__stex_statements_definiendum_args:n { #1 }
4053   \stex_get_symbol:n { #2 }
4054   \str_set:Nx \l_tmpa_str {
4055     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4056   }
4057   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4058   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4059 \rustex_if:TF {
4060   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4061     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4062   }
4063 } {
4064   \defemph@uri {
4065     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4066   } { \l_stex_get_symbol_uri_str }
4067 }
4068 }
4069 \stex_deactivate_macro:Nn \Definame {definition-environments}
4070
4071 \NewDocumentCommand \Symname { 0{ } m }{
4072   \stex_symname_args:n { #1 }
4073   \stex_get_symbol:n { #2 }
4074   \str_set:Nx \l_tmpa_str {
4075     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4076   }
4077   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4078   \let\compemph_uri_prev:\compemph@uri
4079   \let\compemph@uri\symrefemph@uri
4080   \exp_args:NNx \use:nn
4081   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
4082     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4083     \l_stex_symname_post_str
4084   ] }
4085   \let\compemph@uri\compemph_uri_prev:
4086 }

```

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

sdefinition

```

4087
4088 \keys_define:nn {stex / sdefinition }{
4089   type      .str_set_x:N = \sdefinitiontype,
4090   id        .str_set_x:N = \sdefinitionid,
4091   name      .str_set_x:N = \sdefinitionname,
4092   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4093   title     .tl_set:N     = \sdefinitiontitle
4094 }
4095 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4096   \str_clear:N \sdefinitiontype
4097   \str_clear:N \sdefinitionid
4098   \str_clear:N \sdefinitionname
4099   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4100   \tl_clear:N \sdefinitiontitle
4101   \keys_set:nn { stex / sdefinition }{ #1 }
4102 }
4103
4104 \NewDocumentEnvironment{sdefinition}{0{}}{
4105   \__stex_statements_sdefinition_args:n{ #1 }
4106   \stex_reactivate_macro:N \definiendum
4107   \stex_reactivate_macro:N \definame
4108   \stex_reactivate_macro:N \Definame

```

```

4109 \stex_if_smsmode:F{
4110   \seq_clear:N \l_tmpa_seq
4111   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4112     \str_if_eq:nnF{ ##1 }{}{
4113       \stex_get_symbol:n { ##1 }
4114       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4115         \l_stex_get_symbol_uri_str
4116       }
4117     }
4118   }
4119   \exp_args:Nnnx
4120   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4121   \str_if_empty:NF \sdefinitiontype {
4122     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4123   }
4124   \clist_set:No \l_tmpa_clist \sdefinitiontype
4125   \tl_clear:N \l_tmpa_tl
4126   \clist_map_inline:Nn \l_tmpa_clist {
4127     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4128       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4129     }
4130   }
4131   \tl_if_empty:NTF \l_tmpa_tl {
4132     \__stex_statements_sdefinition_start:
4133   }{
4134     \l_tmpa_tl
4135   }
4136 }
4137 \str_if_empty:NF \sdefinitionid {
4138   \stex_ref_new_doc_target:n \sdefinitionid
4139 }
4140 \stex_smsmode_do:
4141 }{
4142   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4143   \stex_if_smsmode:F {
4144     \clist_set:No \l_tmpa_clist \sdefinitiontype
4145     \tl_clear:N \l_tmpa_tl
4146     \clist_map_inline:Nn \l_tmpa_clist {
4147       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4148         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4149       }
4150     }
4151     \tl_if_empty:NTF \l_tmpa_tl {
4152       \__stex_statements_sdefinition_end:
4153     }{
4154       \l_tmpa_tl
4155     }
4156     \end{stex_annotate_env}
4157   }
4158 }

```

\stexpatchdefinition

```

4159 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4160   \par\noindent\titllemph{Definition}\tl_if_empty:NF \sdefinitiontitle {

```

```

4161 ~(\sdefinitiontitle)
4162 }~}
4163 }
4164 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4165
4166 \newcommand\stexpatchdefinition[3] [] {
4167   \str_set:Nx \l_tmpa_str{ #1 }
4168   \str_if_empty:NTF \l_tmpa_str {
4169     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4170     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4171   }{
4172     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4173     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4174   }
4175 }

```

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

\inlinedef inline:

```

4176 \keys_define:nn {stex / inlinedef }{
4177   type      .str_set_x:N = \sdefinitiontype,
4178   id        .str_set_x:N = \sdefinitionid,
4179   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4180   name      .str_set_x:N = \sdefinitionname
4181 }
4182 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4183   \str_clear:N \sdefinitiontype
4184   \str_clear:N \sdefinitionid
4185   \str_clear:N \sdefinitionname
4186   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4187   \keys_set:nn { stex / inlinedef }{ #1 }
4188 }
4189 \NewDocumentCommand \inlinedef { 0{} m } {
4190   \begingroup
4191   \__stex_statements_inlinedef_args:n{ #1 }
4192   \stex_reactivate_macro:N \definiendum
4193   \stex_reactivate_macro:N \definame
4194   \stex_reactivate_macro:N \Definame
4195   \str_if_empty:NF \sdefinitionid {
4196     \stex_ref_new_doc_target:n \sdefinitionid
4197   }
4198   \stex_if_smsmode:TF{
4199     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4200   }{
4201     \seq_clear:N \l_tmpa_seq
4202     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4203       \str_if_eq:nnF{ ##1 }{ }{
4204         \stex_get_symbol:n { ##1 }
4205         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4206           \l_stex_get_symbol_uri_str
4207         }
4208       }
4209     }
4210     \exp_args:Nnx

```

```

4211 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {},,}{
4212 \str_if_empty:NF \sdefinitiontype {
4213 \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4214 }
4215 #2
4216 \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4217 }
4218 }
4219 \endgroup
4220 \stex_smsmode_do:
4221 }

```

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

33.2 Assertions

sassertion

```

4222
4223 \keys_define:nn {stex / sassertion }{
4224 type .str_set_x:N = \sassertiontype,
4225 id .str_set_x:N = \sassertionid,
4226 title .tl_set:N = \sassertiontitle ,
4227 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4228 name .str_set_x:N = \sassertionname
4229 }
4230 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4231 \str_clear:N \sassertiontype
4232 \str_clear:N \sassertionid
4233 \str_clear:N \sassertionname
4234 \clist_clear:N \l__stex_statements_sassertion_for_clist
4235 \tl_clear:N \sassertiontitle
4236 \keys_set:nn { stex / sassertion }{ #1 }
4237 }
4238
4239 %\tl_new:N \g__stex_statements_aftergroup_tl
4240
4241 \NewDocumentEnvironment{sassertion}{0{}}{
4242 \__stex_statements_sassertion_args:n{ #1 }
4243 \stex_if_smsmode:F {
4244 \seq_clear:N \l_tmpa_seq
4245 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4246 \str_if_eq:nnF{ ##1 }{ }{
4247 \stex_get_symbol:n { ##1 }
4248 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4249 \l_stex_get_symbol_uri_str
4250 }
4251 }
4252 }
4253 \exp_args:Nnnx
4254 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {},,}
4255 \str_if_empty:NF \sassertiontype {
4256 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4257 }

```

```

4258 \clist_set:No \l_tmpa_clist \sassertiontype
4259 \tl_clear:N \l_tmpa_tl
4260 \clist_map_inline:Nn \l_tmpa_clist {
4261   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4262     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4263   }
4264 }
4265 \tl_if_empty:NTF \l_tmpa_tl {
4266   \__stex_statements_sassertion_start:
4267 }{
4268   \l_tmpa_tl
4269 }
4270 }
4271 \str_if_empty:NTF \sassertionid {
4272   \str_if_empty:NF \sassertionname {
4273     \stex_ref_new_doc_target:n {}
4274   }
4275 } {
4276   \stex_ref_new_doc_target:n \sassertionid
4277 }
4278 \stex_smsmode_do:
4279 }{
4280   \str_if_empty:NF \sassertionname {
4281     \symdecl*{\sassertionname}
4282     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4283   }
4284   \stex_if_smsmode:F {
4285     \clist_set:No \l_tmpa_clist \sassertiontype
4286     \tl_clear:N \l_tmpa_tl
4287     \clist_map_inline:Nn \l_tmpa_clist {
4288       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4289         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4290       }
4291     }
4292     \tl_if_empty:NTF \l_tmpa_tl {
4293       \__stex_statements_sassertion_end:
4294     }{
4295       \l_tmpa_tl
4296     }
4297     \end{stex_annotate_env}
4298   }
4299 }

```

\stexpatchassertion

```

4300
4301 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4302   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4303     (\sassertiontitle)
4304   }~}
4305 }
4306 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4307
4308 \newcommand\stexpatchassertion[3] [] {
4309   \str_set:Nx \l_tmpa_str{ #1 }

```

```

4310 \str_if_empty:NTF \l_tmpa_str {
4311   \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4312   \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4313 }{
4314   \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4315   \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4316 }
4317 }

```

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

\inlineass inline:

```

4318 \keys_define:nn {stex / inlineass }{
4319   type      .str_set_x:N = \sassertiontype,
4320   id        .str_set_x:N = \sassertionid,
4321   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4322   name      .str_set_x:N = \sassertionname
4323 }
4324 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4325   \str_clear:N \sassertiontype
4326   \str_clear:N \sassertionid
4327   \str_clear:N \sassertionname
4328   \clist_clear:N \l__stex_statements_sassertion_for_clist
4329   \keys_set:nn { stex / inlineass }{ #1 }
4330 }
4331 \NewDocumentCommand \inlineass { 0{} m } {
4332   \begingroup
4333   \__stex_statements_inlineass_args:n{ #1 }
4334   \str_if_empty:NTF \sassertionid {
4335     \str_if_empty:NF \sassertionname {
4336       \stex_ref_new_doc_target:n { }
4337     }
4338   } {
4339     \stex_ref_new_doc_target:n \sassertionid
4340   }
4341 }
4342 \stex_if_smsmode:TF{
4343   \str_if_empty:NF \sassertionname {
4344     \symdecl*{\sassertionname}
4345     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4346   }
4347 }{
4348   \seq_clear:N \l_tmpa_seq
4349   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4350     \str_if_eq:nnF{ ##1 }{ }{
4351       \stex_get_symbol:n { ##1 }
4352       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4353         \l_stex_get_symbol_uri_str
4354       }
4355     }
4356   }
4357   \exp_args:Nnx
4358   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4359     \str_if_empty:NF \sassertiontype {

```



```

4360     \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4361   }
4362   #2
4363   \str_if_empty:NF \sassertionname {
4364     \symdecl*{\sassertionname}
4365     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4366   }
4367 }
4368 }
4369 \endgroup
4370 \stex_smsmode_do:
4371 }

```

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

33.3 Examples

`sexample`

```

4372
4373 \keys_define:nn {stex / sexample }{
4374   type      .str_set_x:N = \exampletype,
4375   id        .str_set_x:N = \sexampleid,
4376   title     .tl_set:N    = \sexampletitle,
4377   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4378 }
4379 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4380   \str_clear:N \sexampletype
4381   \str_clear:N \sexampleid
4382   \tl_clear:N \sexampletitle
4383   \clist_clear:N \l__stex_statements_sexample_for_clist
4384   \keys_set:nn { stex / sexample }{ #1 }
4385 }
4386
4387 \NewDocumentEnvironment{sexample}{0{}}{
4388   \__stex_statements_sexample_args:n{ #1 }
4389   \stex_if_smsmode:F {
4390     \seq_clear:N \l_tmpa_seq
4391     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4392       \str_if_eq:nnF{ ##1 }{}{
4393         \stex_get_symbol:n { ##1 }
4394         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4395           \l_stex_get_symbol_uri_str
4396         }
4397       }
4398     }
4399     \exp_args:Nnnx
4400     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4401     \str_if_empty:NF \sexampletype {
4402       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4403     }
4404     \clist_set:Nn \l_tmpa_clist \sexampletype
4405     \tl_clear:N \l_tmpa_tl
4406     \clist_map_inline:Nn \l_tmpa_clist {

```

```

4407     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4408         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4409     }
4410 }
4411 \tl_if_empty:NTF \l_tmpa_tl {
4412     \__stex_statements_sexample_start:
4413 }{
4414     \l_tmpa_tl
4415 }
4416 }
4417 \str_if_empty:NF \sexampleid {
4418     \stex_ref_new_doc_target:n \sexampleid
4419 }
4420 \stex_smsmode_do:
4421 ){
4422     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4423     \stex_if_smsmode:F {
4424         \clist_set:Nn \l_tmpa_clist \sexampletype
4425         \tl_clear:N \l_tmpa_tl
4426         \clist_map_inline:Nn \l_tmpa_clist {
4427             \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4428                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4429             }
4430         }
4431         \tl_if_empty:NTF \l_tmpa_tl {
4432             \__stex_statements_sexample_end:
4433         }{
4434             \l_tmpa_tl
4435         }
4436         \end{stex_annotate_env}
4437     }
4438 }

```

`\stexpatchexample`

```

4439
4440 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4441     \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampletitle {
4442         (\sexampletitle)
4443     }~}
4444 }
4445 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4446
4447 \newcommand\stexpatchexample[3][[] {
4448     \str_set:Nx \l_tmpa_str{ #1 }
4449     \str_if_empty:NTF \l_tmpa_str {
4450         \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4451         \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4452     }{
4453         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4454         \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4455     }
4456 }

```

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

`\inlineex` inline:

```

4457 \keys_define:nn {stex / inlineex }{
4458   type      .str_set_x:N = \sexamplotype,
4459   id        .str_set_x:N = \sexampleid,
4460   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4461   name      .str_set_x:N = \sexamplename
4462 }
4463 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4464   \str_clear:N \sexamplotype
4465   \str_clear:N \sexampleid
4466   \str_clear:N \sexamplename
4467   \clist_clear:N \l__stex_statements_sexample_for_clist
4468   \keys_set:nn { stex / inlineex }{ #1 }
4469 }
4470 \NewDocumentCommand \inlineex { 0{} m } {
4471   \begingroup
4472   \__stex_statements_inlineex_args:n{ #1 }
4473   \str_if_empty:NF \sexampleid {
4474     \stex_ref_new_doc_target:n \sexampleid
4475   }
4476   \stex_if_smsmode:TF{
4477     \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4478   }{
4479     \seq_clear:N \l_tmpa_seq
4480     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4481       \str_if_eq:nnF{ ##1 }{{}{
4482         \stex_get_symbol:n { ##1 }
4483         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4484           \l_stex_get_symbol_uri_str
4485         }
4486       }
4487     }
4488     \exp_args:Nnx
4489     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4490       \str_if_empty:NF \sexamplotype {
4491         \stex_annotate_invisible:nnn{type}{\sexamplotype}{
4492       }
4493       #2
4494       \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4495     }
4496   }
4497   \endgroup
4498   \stex_smsmode_do:
4499 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4500 \keys_define:nn { stex / sparagraph } {
4501   id      .str_set_x:N = \sparagraphid ,
4502   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,

```

```

4503 type      .str_set_x:N    = \sparagraphtype ,
4504 for        .clist_set:N    = \l__stex_statements_sparagraph_for_clist ,
4505 from        .tl_set:N      = \sparagraphfrom ,
4506 to          .tl_set:N      = \sparagraphto ,
4507 start       .tl_set:N      = \l_stex_sparagraph_start_tl ,
4508 name        .str_set:N      = \sparagraphname
4509 }
4510
4511 \cs_new_protected:Nn \stex_sparagraph_args:n {
4512   \tl_clear:N \l_stex_sparagraph_title_tl
4513   \tl_clear:N \sparagraphfrom
4514   \tl_clear:N \sparagraphto
4515   \tl_clear:N \l_stex_sparagraph_start_tl
4516   \str_clear:N \sparagraphid
4517   \str_clear:N \sparagraphtype
4518   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4519   \str_clear:N \sparagraphname
4520   \keys_set:nn { stex / sparagraph }{ #1 }
4521 }
4522 \newif\if@in@omtext\@in@omtextfalse
4523
4524 \NewDocumentEnvironment {sparagraph} { 0{} } {
4525   \stex_sparagraph_args:n { #1 }
4526   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4527     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4528   }{
4529     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4530   }
4531   \@in@omtexttrue
4532   \stex_if_smsmode:F {
4533     \seq_clear:N \l_tmpa_seq
4534     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4535       \str_if_eq:nnF{ ##1 }{ }{
4536         \stex_get_symbol:n { ##1 }
4537         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4538           \l_stex_get_symbol_uri_str
4539         }
4540       }
4541     }
4542     \exp_args:Nnnx
4543     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4544     \str_if_empty:NF \sparagraphtype {
4545       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4546     }
4547     \str_if_empty:NF \sparagraphfrom {
4548       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4549     }
4550     \str_if_empty:NF \sparagraphto {
4551       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4552     }
4553     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4554     \tl_clear:N \l_tmpa_tl
4555     \clist_map_inline:Nn \sparagraphtype {
4556       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{

```

```

4557     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4558   }
4559 }
4560 \tl_if_empty:NTF \l_tmpa_tl {
4561   \__stex_statements_sparagraph_start:
4562 }{
4563   \l_tmpa_tl
4564 }
4565 }
4566 \clist_set:No \l_tmpa_clist \sparagraphtype
4567 \str_if_empty:NTF \sparagraphid {
4568   \str_if_empty:NTF \sparagraphname {
4569     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4570       \stex_ref_new_doc_target:n {}
4571     }
4572   } {
4573     \stex_ref_new_doc_target:n {}
4574   }
4575 } {
4576   \stex_ref_new_doc_target:n \sparagraphid
4577 }
4578 \exp_args:NNx
4579 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4580   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4581     \str_if_eq:nnF{ ##1 }{ }{
4582       \stex_get_symbol:n { ##1 }
4583       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4584     }
4585   }
4586 }
4587 \stex_smsmode_do:
4588 \ignorespacesandpars
4589 }{
4590   \str_if_empty:NF \sparagraphname {
4591     \symdecl*{\sparagraphname}
4592     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4593   }
4594   \stex_if_smsmode:F {
4595     \clist_set:No \l_tmpa_clist \sparagraphtype
4596     \tl_clear:N \l_tmpa_tl
4597     \clist_map_inline:Nn \l_tmpa_clist {
4598       \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
4599         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4600       }
4601     }
4602     \tl_if_empty:NTF \l_tmpa_tl {
4603       \__stex_statements_sparagraph_end:
4604     }{
4605       \l_tmpa_tl
4606     }
4607     \end{stex_annotate_env}
4608   }
4609 }

```

\stexpatchparagraph

```

4610
4611 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4612   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4613     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4614       \titleemph{\l_stex_sparagraph_title_tl}:~
4615     }
4616   }{
4617     \titleemph{\l_stex_sparagraph_start_tl}~
4618   }
4619 }
4620 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4621
4622 \newcommand\stexpatchparagraph[3] [] {
4623   \str_set:Nx \l_tmpa_str{ #1 }
4624   \str_if_empty:NTF \l_tmpa_str {
4625     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4626     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4627   }{
4628     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4629     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4630   }
4631 }
4632
4633 \keys_define:nn { stex / inlinepara } {
4634   id      .str_set:x:N = \sparagraphid ,
4635   type    .str_set:x:N = \sparagraphtype ,
4636   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4637   from    .tl_set:N    = \sparagraphfrom ,
4638   to      .tl_set:N    = \sparagraphto ,
4639   name    .str_set:N    = \sparagraphname
4640 }
4641 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4642   \tl_clear:N \sparagraphfrom
4643   \tl_clear:N \sparagraphto
4644   \str_clear:N \sparagraphid
4645   \str_clear:N \sparagraphtype
4646   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4647   \str_clear:N \sparagraphname
4648   \keys_set:nn { stex / inlinepara }{ #1 }
4649 }
4650 \NewDocumentCommand \inlinepara { O{} m } {
4651   \begingroup
4652     \__stex_statements_inlinepara_args:n{ #1 }
4653     \clist_set:No \l_tmpa_clist \sparagraphtype
4654     \str_if_empty:NTF \sparagraphid {
4655       \str_if_empty:NTF \sparagraphname {
4656         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4657           \stex_ref_new_doc_target:n {}
4658         }
4659       } {
4660         \stex_ref_new_doc_target:n {}
4661       }
4662     } {

```

```

4663 \stex_ref_new_doc_target:n \sparagraphid
4664 }
4665 \stex_if_smsmode:TF{
4666   \str_if_empty:NF \sparagraphname {
4667     \symdecl*{\sparagraphname}
4668     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4669   }
4670 }{
4671   \seq_clear:N \l_tmpa_seq
4672   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4673     \str_if_eq:nnF{ ##1 }{}{
4674       \stex_get_symbol:n { ##1 }
4675       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4676         \l_stex_get_symbol_uri_str
4677       }
4678     }
4679   }
4680   \exp_args:Nnx
4681   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4682     \str_if_empty:NF \sparagraphtype {
4683       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4684     }
4685     \str_if_empty:NF \sparagraphfrom {
4686       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4687     }
4688     \str_if_empty:NF \sparagraphto {
4689       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4690     }
4691     \str_if_empty:NF \sparagraphname {
4692       \symdecl*{\sparagraphname}
4693       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4694     }
4695     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4696       \clist_map_inline:Nn \l_tmpa_seq {
4697         \stex_ref_new_sym_target:n {##1}
4698       }
4699     }
4700     #2
4701   }
4702 }
4703 \endgroup
4704 \stex_smsmode_do:
4705 }
4706

```

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

symboldoc

```

4707 \NewDocumentEnvironment{symboldoc}{ m }{
4708   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4709   \seq_clear:N \l_tmpb_seq
4710   \seq_map_inline:Nn \l_tmpa_seq {
4711     \str_if_eq:nnF{ ##1 }{}{
4712       \stex_get_symbol:n { ##1 }

```

```

4713     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4714         \l_stex_get_symbol_uri_str
4715     }
4716 }
4717 }
4718 \par
4719 \exp_args:Nnnx
4720 \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4721 }{
4722     \end{stex_annotate_env}
4723 }

4724 \end{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

```
4725 \package
4726 \@@=stex_sproof
4727
4728 %%%%%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%%%%
4729
```

34.2 Proofs

We first define some keys for the proof environment.

```
4730 \keys_define:nn { stex / spf } {
4731   id          .str_set:N = \l__stex_sproof_spf_id_str,
4732   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4733   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4734   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4735   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4736   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4737   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4738   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4739   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4740   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4741 }
4742 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4743   \str_clear:N \l__stex_sproof_spf_id_str
4744   \tl_clear:N \l__stex_sproof_spf_display_tl
4745   \tl_clear:N \l__stex_sproof_spf_for_tl
4746   \tl_clear:N \l__stex_sproof_spf_from_tl
4747   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4748   \tl_clear:N \l__stex_sproof_spf_type_tl
4749   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4750 \tl_clear:N \l__stex_sproof_spf_continues_tl
4751 \tl_clear:N \l__stex_sproof_spf_functions_tl
4752 \tl_clear:N \l__stex_sproof_spf_method_tl
4753 \keys_set:nn { stex / spf }{ #1 }
4754 }

```

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

```

4755 \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.

```

4756 \newcount\count_ten
4757 \newenvironment{pst@with@label}[1]{
4758   \edef\pst@label{#1}
4759   \advance\count_ten by 1\relax
4760   \count_ten=1
4761 }{
4762   \advance\count_ten by -1\relax
4763 }

```

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

```

4764 \def\the@pst@label{
4765   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4766 }

```

(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.

```

4767 \keys_define:nn { stex / pstlabel }{
4768   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4769   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4770   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4771 }
4772 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4773 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4774 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4775 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4776 }
4777 \__stex_sproof_pstlabel_args:n {}
4778 \newcommand\setpstlabelstyle[1]{
4779   \__stex_sproof_pstlabel_args:n {#1}
4780 }
4781 \newcommand\setpstlabelstyledefault{%
4782   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4783 }

```

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

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

```

4784 \ExplSyntaxOff
4785 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4786 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4787 \def\pst@make@label@short#1#2{#2}
4788 \def\pst@make@label@empty#1#2{}
4789 \ExplSyntaxOn
4790 \def\pstlabelstyle#1{%
4791   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4792 }%
4793 \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.

```

4794 \def\next@pst@label{%
4795   \global\advance\count\count10 by 1%
4796 }%

```

(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

```

4797 \def\sproof@box{
4798   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4799 }
4800 \def\spf@proofend{\sproof@box}
4801 \def\sproofend{
4802   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4803     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4804   }
4805 }
4806 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4807 \def\spf@proofsketch@kw{Proof Sketch}
4808 \def\spf@proof@kw{Proof}
4809 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4810 \AddToHook{begindocument}{
4811   \ltx@ifpackageloaded{babel}{
4812     \makeatletter
4813     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4814     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4815       \input{sproof-ngerman.ldf}
4816     }
4817     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4818       \input{sproof-finnish.ldf}
4819     }
4820     \clist_if_in:NnT \l_tmpa_clist {french}{
4821       \input{sproof-french.ldf}
4822     }
4823     \clist_if_in:NnT \l_tmpa_clist {russian}{
4824       \input{sproof-russian.ldf}
4825     }
4826     \makeatother
4827   }{}
4828 }

```

`spfsketch`

```

4829 \newcommand\spfsketch[2][]{
4830   \__stex_sproof_spf_args:n{#1}
4831   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4832     \titleemph{
4833       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4834         \spf@proofsketch@kw
4835       }{
4836         \l__stex_sproof_spf_type_tl
4837       }
4838     }:
4839   }
4840   {~#2}
4841   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4842   \sproofend
4843 }

```

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

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

```

4844 \newenvironment{spfeq}[2][]{
4845   \__stex_sproof_spf_args:n{#1}
4846   %\sref@target
4847   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4848     \titleemph{
4849       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4850         \spf@proof@kw
4851       }{

```

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

¹⁵EDNOTE: document above

```

4852         \l__stex_sproof_spf_type_tl
4853     }
4854     }:
4855 }
4856 {-#2}
4857 \begin{displaymath}\begin{array}{rcll}
4858 }{
4859 \end{array}\end{displaymath}
4860 }

```

(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.

```

4861 \newenvironment{spf@proof}[2] []{
4862   \l__stex_sproof_spf_args:n{#1}
4863   %\sref@target
4864   \count_ten=10
4865   \par\noindent
4866   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4867     \titleemph{
4868       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4869         \spf@proof@kw
4870       }{
4871         \l__stex_sproof_spf_type_tl
4872       }
4873     }:
4874   }
4875   {-#2}
4876   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4877   \def\pst@label{}
4878   \newcount\pst@count% initialize the labeling mechanism
4879   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4880 }{
4881   \end{pst@with@label}\end{description}
4882 }
4883 \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4884 \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4885 \newcommand\spfidea[2] []{
4886   \l__stex_sproof_spf_args:n{#1}
4887   \titleemph{
4888     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4889       \l__stex_sproof_spf_type_tl
4890     }:
4891   }-#2
4892   \sproofend
4893 }

```

(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
4894 \newenvironment{spfstep}[1][]{
4895   \_stex_sproof_spf_args:n{#1}
4896   \@in@omtexttrue
4897   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4898     \item[\the@pst@label]
4899   }
4900   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4901     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4902   }
4903   %\sref@label@id{\pst@label}
4904   \ignorespacesandpars
4905 }{
4906   \next@pst@label\ignorespacesandpars
4907 }

```

sproofcomment

```

4908 \newenvironment{sproofcomment}[1][]{
4909   \_stex_sproof_spf_args:n{#1}
4910   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4911     \item[\the@pst@label]
4912   }
4913 }{
4914   \next@pst@label
4915 }

```

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.

```

4916 \newenvironment{subproof}[2][]{
4917   \_stex_sproof_spf_args:n{#1}
4918   \def\@test{#2}
4919   \ifx\@test\empty\else
4920     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4921       \item[\the@pst@label]
4922     }{#2}
4923   \fi
4924   \begin{pst@with@label}{\pst@label,\number\count_ten}
4925 }{
4926   \end{pst@with@label}\next@pst@label
4927 }

```

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

```

4928 \newenvironment{spfcases}[2][]{
4929   \def\@test{#1}
4930   \ifx\@test\empty
4931     \begin{subproof}[method=by-cases]{#2}

```

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

```

4932 \else
4933   \begin{subproof}[#1,method=by-cases]{#2}
4934 \fi
4935 }{
4936   \end{subproof}
4937 }

```

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

```

4938 \newenvironment{spfcase}[2] []{
4939   \__stex_sproof_spf_args:n{#1}
4940   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4941     \item[\the@pst@label]
4942   }
4943   \def\@test{#2}
4944   \ifx\@test\@empty
4945   \else
4946     {\titleemph{#2}:~}
4947   \fi
4948   \begin{pst@with@label}{\pst@label,\number\count_ten}
4949 }{
4950   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4951     \sproofend
4952   }
4953   \end{pst@with@label}
4954   \next@pst@label
4955 }

```

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

```

4956 \newcommand\spfcasesketch[3] []{
4957   \__stex_sproof_spf_args:n{#1}
4958   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4959     \item[\the@pst@label]
4960   }
4961   \def\@test{#2}
4962   \ifx\@test\@empty
4963   \else
4964     {\titleemph{#2}:~}
4965   \fi#3
4966   \next@pst@label
4967 }%

```

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.

```

4968 \keys_define:nn { stex / just }{
4969   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4970   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4971   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4972   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4973 }

```

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

`justification`

4974 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4975 `\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.

4976 `\newcommand\justarg[2] [] {#2}`

4977 `\</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

```
4978 <*package>
4979
4980 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4981
4982 <@@=stex_others>
    Warnings and error messages
4983 % None

\MSC Math subject classifier

4984 \NewDocumentCommand \MSC {m} {
4985 % TODO
4986 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4987 \@ifpackageloaded{tikzinput}{
4988 \RequirePackage{stex-tikzinput}
4989 }{}
4990 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4991 \*package>
4992 \@@=stex_modules>
4993
4994 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4995
4996 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4997 \begingroup
4998 \stex_module_setup:nn{
4999   ns=\c_stex_metatheory_ns_str,
5000   meta=NONE
5001 }{Metatheory}
5002 \stex_reactivate_macro:N \symdecl
5003 \stex_reactivate_macro:N \notation
5004 \stex_reactivate_macro:N \symdef
5005 \ExplSyntaxOff
5006 \csname stex_suppress_html:n\endcsname{
5007   % is-a (a:A, a \in A, a is an A, etc.)
5008   \symdecl[args=ai]{isa}
5009   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
5010   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
5011   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
5012
5013   % bind (\forall, \Pi, \lambda etc.)
5014   \symdecl[args=Bi]{bind}
5015   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
5016   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
5017   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5018
5019   % dummy variable
5020   \symdecl{dummyvar}
5021   \notation[underscore]{dummyvar}{\comp\_}
5022   \notation[dot]{dummyvar}{\comp\cdot}
5023   \notation[dash]{dummyvar}{\comp{\rm --}}
5024
5025   %fromto (function space, Hom-set, implication etc.)
```

```

5026 \symdecl[args=ai]{fromto}
5027 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
5028 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
5029
5030 % mapto (lambda etc.)
5031 \%symdecl[args=Bi]{mapto}
5032 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
5033 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5034 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5035
5036 % function/operator application
5037 \symdecl[args=ia]{apply}
5038 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5039 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
5040
5041 % ‘type’ of all collections (sets, classes, types, kinds)
5042 \symdecl{collection}
5043 \notation[U]{collection}{\comp{\mathcal{U}}}
5044 \notation[set]{collection}{\comp{\textsf{Set}}}
5045
5046 % sequences
5047 \symdecl[args=1]{seqtype}
5048 \notation[kleene]{seqtype}{#1^{\comp\ast}}
5049
5050 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
5051 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
5052
5053 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
5054 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
5055 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
5056
5057 % letin (‘let’, local definitions, variable substitution)
5058 \symdecl[args=bii]{letin}
5059 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
5060 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
5061 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
5062
5063 % structures
5064 \symdecl*[args=1]{module-type}
5065 \notation{module-type}{\mathtt{MOD} #1}
5066 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
5067 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5068
5069 }
5070 \ExplSyntaxOn
5071 \stex_add_to_current_module:n{
5072   \let\nappa\apply
5073   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5074   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5075   \def\livar{\csname sequence-index\endcsname[li]}
5076   \def\uivar{\csname sequence-index\endcsname[ui]}
5077   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5078   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5079   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```

```
5080 }  
5081 \__stex_modules_end_module:  
5082 \endgroup  
5083 </package>
```

Chapter 37

Tikzinput Implementation

```
5084 <*package>
5085
5086 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5087
5088 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5089 \RequirePackage{l3keys2e}
5090
5091 \keys_define:nn { tikzinput } {
5092   image .bool_set:N = \c_tikzinput_image_bool,
5093   image .default:n = false ,
5094   unknown .code:n = {}
5095 }
5096
5097 \ProcessKeysOptions { tikzinput }
5098
5099 \bool_if:NTF \c_tikzinput_image_bool {
5100   \RequirePackage{graphicx}
5101
5102   \providecommand\usetikzlibrary[]{}
5103   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5104 }{
5105   \RequirePackage{tikz}
5106   \RequirePackage{standalone}
5107
5108   \newcommand \tikzinput [2] [] {
5109     \setkeys{Gin}{#1}
5110     \ifx \Gin@ewidth \Gin@exclamation
5111       \ifx \Gin@eheight \Gin@exclamation
5112         \input { #2 }
5113       \else
5114         \resizebox{!}{ \Gin@eheight }{
5115           \input { #2 }
5116         }
5117       \fi
5118     \else
5119       \ifx \Gin@eheight \Gin@exclamation
5120         \resizebox{ \Gin@ewidth }{!}{
5121           \input { #2 }
```

```

5122     }
5123     \else
5124         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5125             \input { #2 }
5126         }
5127     \fi
5128 \fi
5129 }
5130 }
5131
5132 \newcommand \ctikzinput [2] [] {
5133     \begin{center}
5134         \tikzinput [1] {#2}
5135     \end{center}
5136 }
5137
5138 \@ifpackageloaded{stex}{
5139     \RequirePackage{stex-tikzinput}
5140 }{}
5141
5142 </package>
5143 <*stex>
5144 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5145 \RequirePackage{stex}
5146 \RequirePackage{tikzinput}
5147
5148 \newcommand\mhtikzinput [2] [] {%
5149     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5150     \stex_in_repository:nn\Gin@mhrepos{
5151         \tikzinput [1]{\mhpath{##1}{#2}}
5152     }
5153 }
5154 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5155 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The 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.

```
5156 \*cls)
5157 \@@=document_structure)
5158 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5159 \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`

```
5160 \keys_define:nn{ document-structure / pkg }{
5161   class      .str_set_x:N = \c_document_structure_class_str,
5162   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5163   report     .code:n      = {
5164     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5165     \str_set:Nn \c_document_structure_class_str {report}
5166   },
5167   book       .code:n      = {
5168     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5169     \str_set:Nn \c_document_structure_class_str {book}
5170   },
5171   bookpart   .code:n      = {
5172     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5173     \str_set:Nn \c_document_structure_class_str {book}
5174     \str_set:Nn \c_document_structure_topsect_str {chapter}
5175   },
```

```

5176 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5177 unknown     .code:n      = {
5178   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5179 }
5180 }
5181 \ProcessKeysOptions{ document-structure / pkg }
5182 \str_if_empty:NT \c_document_structure_class_str {
5183   \str_set:Nn \c_document_structure_class_str {article}
5184 }
5185 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5186   {\c_document_structure_class_str}
5187

```

38.3 Beefing up the document environment

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

```

5188 \RequirePackage{document-structure}
5189 \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

```

5190 \keys_define:nn { document-structure / document }{
5191   id .str_set_x:N = \c_document_structure_document_id_str
5192 }
5193 \let\__document_structure_orig_document=\document
5194 \renewcommand{\document}[1][]{
5195   \keys_set:nn{ document-structure / document }{ #1 }
5196   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5197   \__document_structure_orig_document
5198 }

```

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

```

5199 }
5200 </cls>

```

38.4 Implementation: document-structure Package

```

5201 <*package>
5202 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5203 \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


```

5204
5205 \keys_define:nn{ document-structure / pkg }{
5206   class      .str_set_x:N = \c_document_structure_class_str,
5207   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5208   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5209 }
5210 \ProcessKeysOptions{ document-structure / pkg }
5211 \str_if_empty:NT \c_document_structure_class_str {
5212   \str_set:Nn \c_document_structure_class_str {article}
5213 }
5214 \str_if_empty:NT \c_document_structure_topsect_str {
5215   \str_set:Nn \c_document_structure_topsect_str {section}
5216 }

```

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

```

5217 \RequirePackage{xspace}
5218 \RequirePackage{comment}
5219 \AddToHook{begindocument}{
5220   \ltx@ifpackageloaded{babel}{
5221     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5222     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5223       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5224     }
5225   }{}
5226 }

```

`\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.

```

5227 \int_new:N \l_document_structure_section_level_int
5228 \str_case:VnF \c_document_structure_topsect_str {
5229   {part}}{
5230     \int_set:Nn \l_document_structure_section_level_int {0}
5231   }
5232   {chapter}}{
5233     \int_set:Nn \l_document_structure_section_level_int {1}
5234   }
5235 }{
5236   \str_case:VnF \c_document_structure_class_str {
5237     {book}}{
5238       \int_set:Nn \l_document_structure_section_level_int {0}
5239     }
5240     {report}}{
5241       \int_set:Nn \l_document_structure_section_level_int {0}
5242     }
5243   }{
5244     \int_set:Nn \l_document_structure_section_level_int {2}
5245   }
5246 }

```

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

```
5247 \def\current@section@level{document}%
5248 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5249 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5250 \cs_new_protected:Npn \skipomgroup {
5251   \ifcase\l_document_structure_section_level_int
5252   \or\stepcounter{part}
5253   \or\stepcounter{chapter}
5254   \or\stepcounter{section}
5255   \or\stepcounter{subsection}
5256   \or\stepcounter{subsubsection}
5257   \or\stepcounter{paragraph}
5258   \or\stepcounter{subparagraph}
5259   \fi
5260 }
```

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

`blindomgroup`

```
5261 \newcommand\at@begin@blindomgroup[1]{%
5262 \newenvironment{blindomgroup}
5263 {
5264   \int_incr:N\l_document_structure_section_level_int
5265   \at@begin@blindomgroup\l_document_structure_section_level_int
5266 }{}}
```

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

```
5267 \newcommand\omgroup@nonum[2]{
5268   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5269   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5270 }
```

(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.

```
5271 \newcommand\omgroup@num[2]{
```

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

```

5272 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5273   \@nameuse{#1}{#2}
5274 }{
5275   \cs_if_exist:NTF\rdfmata@sectioning{
5276     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5277   }{
5278     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5279   }
5280 }
5281 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5282 }

```

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

omgroup

```

5283 \keys_define:nn { document-structure / omgroup }{
5284   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5285   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5286   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5287   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5288   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5289   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5290   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5291   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5292   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5293   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5294 }
5295 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5296   \str_clear:N \l__document_structure_omgroup_id_str
5297   \str_clear:N \l__document_structure_omgroup_date_str
5298   \clist_clear:N \l__document_structure_omgroup_creators_clist
5299   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5300   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5301   \tl_clear:N \l__document_structure_omgroup_type_tl
5302   \tl_clear:N \l__document_structure_omgroup_short_tl
5303   \tl_clear:N \l__document_structure_omgroup_display_tl
5304   \tl_clear:N \l__document_structure_omgroup_intro_tl
5305   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5306   \keys_set:nn { document-structure / omgroup } { #1 }
5307 }

```

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.

```

5308 \newif\if@mainmatter\@mainmattertrue
5309 \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.

```

5310 \keys_define:nn { document-structure / sectioning }{
5311   name .str_set_x:N = \l__document_structure_sect_name_str ,
5312   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5313   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5314   clear .default:n = {true} ,
5315   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5316   num      .default:n      = {true}
5317 }
5318 \cs_new_protected:Nn \__document_structure_sect_args:n {
5319   \str_clear:N \l__document_structure_sect_name_str
5320   \str_clear:N \l__document_structure_sect_ref_str
5321   \bool_set_false:N \l__document_structure_sect_clear_bool
5322   \bool_set_false:N \l__document_structure_sect_num_bool
5323   \keys_set:nn { document-structure / sectioning } { #1 }
5324 }
5325 \newcommand\omdoc@sectioning[3][]{
5326   \__document_structure_sect_args:n {#1}
5327   \let\omdoc@sect@name\l__document_structure_sect_name_str
5328   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5329   \if@mainmatter% numbering not overridden by frontmatter, etc.
5330     \bool_if:NTF \l__document_structure_sect_num_bool {
5331       \omgroup@num{#2}{#3}
5332     }{
5333       \omgroup@nonum{#2}{#3}
5334     }
5335     \def\current@section@level{\omdoc@sect@name}
5336   \else
5337     \omgroup@nonum{#2}{#3}
5338   \fi
5339 }% 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.

```

5340 \newcommand\omgroup@redefine@addtocontents[1]{%
5341 %\edef\__document_structureimport{#1}%
5342 %\@for\@I:=\__document_structureimport\do{%
5343 %\edef\@path{\csname module@\@I @path\endcsname}%
5344 %\@ifundefined{tf@toc}\relax%
5345 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5346 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5347 %\def\addcontentsline##1##2##3{%
5348 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5349 %\else% hyperref.sty not loaded
5350 %\def\addcontentsline##1##2##3{%
5351 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5352 %\fi
5353 }% hypreref.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.

```

5354 \newenvironment{omgroup}[2][]{% keys, title
5355 {
5356   \__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.

```

5357 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5358   \omgroup@redefine@addtocontents{
5359     \@ifundefined{module@id}\used@modules%

```

```

5360     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5361   }
5362 }

```

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

```

5363 \int_incr:N\l_document_structure_section_level_int
5364 \ifcase\l_document_structure_section_level_int
5365   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5366   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5367   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5368   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5369   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5370   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5371   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5372 \fi
5373 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5374 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5375   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5376 }
5377 }% for customization
5378 {}

```

and finally, we localize the sections

```

5379 \newcommand\omdoc@part@kw{Part}
5380 \newcommand\omdoc@chapter@kw{Chapter}
5381 \newcommand\omdoc@section@kw{Section}
5382 \newcommand\omdoc@subsection@kw{Subsection}
5383 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5384 \newcommand\omdoc@paragraph@kw{paragraph}
5385 \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`

```

5386 \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).

```

5387 \cs_if_exist:NTF\frontmatter{
5388   \let\__document_structure_orig_frontmatter\frontmatter
5389   \let\frontmatter\relax
5390 }{
5391   \tl_set:Nn\__document_structure_orig_frontmatter{
5392     \clearpage
5393     \@mainmatterfalse
5394     \pagenumbering{roman}

```

```

5395 }
5396 }
5397 \cs_if_exist:NTF\backmatter{
5398   \let\__document_structure_orig_backmatter\backmatter
5399   \let\backmatter\relax
5400 }{
5401   \tl_set:Nn\__document_structure_orig_backmatter{
5402     \clearpage
5403     \@mainmatterfalse
5404     \pagenumbering{roman}
5405   }
5406 }

```

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.

```

5407 \newenvironment{frontmatter}{
5408   \__document_structure_orig_frontmatter
5409 }{
5410   \cs_if_exist:NTF\mainmatter{
5411     \mainmatter
5412   }{
5413     \clearpage
5414     \@mainmattertrue
5415     \pagenumbering{arabic}
5416   }
5417 }

```

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

```

5418 \newenvironment{backmatter}{
5419   \__document_structure_orig_backmatter
5420 }{
5421   \cs_if_exist:NTF\mainmatter{
5422     \mainmatter
5423   }{
5424     \clearpage
5425     \@mainmattertrue
5426     \pagenumbering{arabic}
5427   }
5428 }

```

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

```

5429 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5430 \def \c__document_structure_document_str{document}
5431 \newcommand\afterprematurestop{}
5432 \def\prematurestop@endomgroup{
5433   \unless\ifx\@currenvir\c__document_structure_document_str
5434     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
5435     \expandafter\prematurestop@endomgroup

```

```

5436 \fi
5437 }
5438 \providecommand\prematurestop{
5439 \message{Stopping~sTeX~processing~prematurely}
5440 \prematurestop@endumgroup
5441 \afterprematurestop
5442 \end{document}
5443 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5444 \RequirePackage{etoolbox}
5445 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5446 \newrobustcmd\useSGvar[1]{%
5447 \@ifundefined{sTeX@Gvar@#1}
5448 {\PackageError{document-structure}
5449 {The sTeX Global variable #1 is undefined}
5450 {set it with \protect\setSGvar}}
5451 \@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.

```

5452 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5453 \@ifundefined{sTeX@Gvar@#1}
5454 {\PackageError{document-structure}
5455 {The sTeX Global variable #1 is undefined}
5456 {set it with \protect\setSGvar}}
5457 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 39

NotesSlides – Implementation

39.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
5458 \*cls)
5459 \@@=notesslides)
5460 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5461 \RequirePackage{l3keys2e,expl-keystr-compatible}
5462
5463 \keys_define:nn{notesslides / cls}{
5464   class .code:n = {
5465     \PassOptionsToClass{\CurrentOption}{document-structure}
5466     \str_if_eq:nnT{#1}{book}{
5467       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5468     }
5469     \str_if_eq:nnT{#1}{report}{
5470       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5471     }
5472   },
5473   notes .bool_set:N = \c__notesslides_notes_bool ,
5474   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5475   unknown .code:n = {
5476     \PassOptionsToClass{\CurrentOption}{document-structure}
5477     \PassOptionsToClass{\CurrentOption}{beamer}
5478     \PassOptionsToPackage{\CurrentOption}{notesslides}
5479   }
5480 }
5481 \ProcessKeysOptions{ notesslides / cls }
5482 \bool_if:NTF \c__notesslides_notes_bool {
5483   \PassOptionsToPackage{notes=true}{notesslides}
5484 }{
5485   \PassOptionsToPackage{notes=false}{notesslides}
5486 }
5487 \</cls)
```


now we do the same for the notesslides package.

```

5488 <*package>
5489 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5490 \RequirePackage{l3keys2e,expl-keystr-compat}
5491
5492 \keys_define:nn{notesslides / pkg}{
5493   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5494   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5495   notes        .bool_set:N = \c__notesslides_notes_bool ,
5496   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5497   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5498   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5499   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5500   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5501   unknown      .code:n      = {
5502     \PassOptionsToClass{\CurrentOption}{stex}
5503     \PassOptionsToClass{\CurrentOption}{tikzinput}
5504   }
5505 }
5506 \ProcessKeysOptions{ notesslides / pkg }
5507 \newif\ifnotes
5508 \bool_if:NTF \c__notesslides_notes_bool {
5509   \notesttrue
5510 }{
5511   \notesfalse
5512 }
5513

```

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

```

5514 \str_if_empty:NTF \c__notesslides_topsect_str {
5515   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5516 }{
5517   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5518 }
5519 </package>

```

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

```

5520 <*cls>
5521 \bool_if:NTF \c__notesslides_notes_bool {
5522   \LoadClass{document-structure}
5523 }{
5524   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5525   \newcounter{Item}
5526   \newcounter{paragraph}
5527   \newcounter{subparagraph}
5528   \newcounter{Hfootnote}
5529   \RequirePackage{document-structure}
5530 }

```

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

```

5531 \RequirePackage{notesslides}
5532 </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.

```

5533 \*package>
5534 \bool_if:NT \c__notesslides_notes_bool {
5535   \RequirePackage{a4wide}
5536   \RequirePackage{marginnote}
5537   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5538   \RequirePackage{mdframed}
5539   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5540   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5541 }
5542 \RequirePackage{stex-tikzinput}
5543 \RequirePackage{etoolbox}
5544 \RequirePackage{amssymb}
5545 \RequirePackage{amsmath}
5546 \RequirePackage{comment}
5547 \RequirePackage{textcomp}
5548 \RequirePackage{url}
5549 \RequirePackage{graphicx}
5550 \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

```

5551 \bool_if:NT \c__notesslides_notes_bool {
5552   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5553 }

```

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

```

5554 \newcounter{slide}
5555 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5556 \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.

```

5557 \bool_if:NTF \c__notesslides_notes_bool {
5558   \renewenvironment{note}{\ignorespaces}{}
5559 }{
5560   \excludecomment{note}
5561 }

```

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.

```
5562 \bool_if:NT \c__notesslides_notes_bool {
5563   \newlength{\slideframewidth}
5564   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5565 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5566   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5567     \bool_set_true:N #1
5568   }{
5569     \bool_set_false:N #1
5570   }
5571 }
5572 \keys_define:nn{notesslides / frame}{
5573   label .str_set_x:N = \l__notesslides_frame_label_str,
5574   allowframebreaks .code:n = {
5575     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5576   },
5577   allowdisplaybreaks .code:n = {
5578     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5579   },
5580   fragile .code:n = {
5581     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5582   },
5583   shrink .code:n = {
5584     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5585   },
5586   squeeze .code:n = {
5587     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5588   },
5589   t .code:n = {
5590     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5591   },
5592 }
5593 \cs_new_protected:Nn \__notesslides_frame_args:n {
5594   \str_clear:N \l__notesslides_frame_label_str
5595   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5596   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5597   \bool_set_true:N \l__notesslides_frame_fragile_bool
5598   \bool_set_true:N \l__notesslides_frame_shrink_bool
5599   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5600   \bool_set_true:N \l__notesslides_frame_t_bool
5601   \keys_set:nn { notesslides / frame }{ #1 }
5602 }
```

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

```
5603 \renewenvironment{frame}[1][]{
5604   \__notesslides_frame_args:n{#1}
5605   \sffamily
5606   \stepcounter{slide}
5607   \def\@currentlabel{\theslide}
5608   \str_if_empty:NF \l__notesslides_frame_label_str {
5609     \label{\l__notesslides_frame_label_str}
```

5610 }

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

```
5611 \def\itemize@level{outer}
5612 \def\itemize@outer{outer}
5613 \def\itemize@inner{inner}
5614 \renewcommand\newpage{\addtocounter{framenum}{1}}
5615 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5616 \renewenvironment{itemize}{
5617   \ifx\itemize@level\itemize@outer
5618     \def\itemize@label{\$ \rhd$}
5619   \fi
5620   \ifx\itemize@level\itemize@inner
5621     \def\itemize@label{\$ \scriptstyle \rhd$}
5622   \fi
5623   \begin{list}
5624     {\itemize@label}
5625     {\setlength{\labelsep}{.3em}
5626      \setlength{\labelwidth}{.5em}
5627      \setlength{\leftmargin}{1.5em}
5628     }
5629   \edef\itemize@level{\itemize@inner}
5630 }{
5631   \end{list}
5632 }
```

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

```
5633 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5634 }{
5635   \medskip\miko@slidelabel\end{mdframed}
5636 }
```

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

`\frametitle`

```
5637 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5638 }
```

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

EdN:21

`\pause` 21

```
5639 \bool_if:NT \c__notesslides_notes_bool {
5640   \newcommand\pause{}
5641 }
```

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

`nparagraph`

```
5642 \bool_if:NTF \c__notesslides_notes_bool {
5643   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}\end{sparagraph}}
5644 }{
5645   \excludecomment{nparagraph}
5646 }
```

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

nomgroup

```
5647 \bool_if:NTF \c__notesslides_notes_bool {  
5648   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
5649 }{  
5650   \excludecomment{nomgroup}  
5651 }
```

ndefinition

```
5652 \bool_if:NTF \c__notesslides_notes_bool {  
5653   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}  
5654 }{  
5655   \excludecomment{ndefinition}  
5656 }
```

nassertion

```
5657 \bool_if:NTF \c__notesslides_notes_bool {  
5658   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}  
5659 }{  
5660   \excludecomment{nassertion}  
5661 }
```

nsproof

```
5662 \bool_if:NTF \c__notesslides_notes_bool {  
5663   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
5664 }{  
5665   \excludecomment{nproof}  
5666 }
```

nexample

```
5667 \bool_if:NTF \c__notesslides_notes_bool {  
5668   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}  
5669 }{  
5670   \excludecomment{nexample}  
5671 }
```

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

```
5672 \def\inputref@preskip{\smallskip}  
5673 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5674 \let\orig@inputref\inputref  
5675 \def\inputref{\@ifstar\ninputref\orig@inputref}  
5676 \newcommand\ninputref[2] [] {  
5677   \bool_if:NT \c__notesslides_notes_bool {  
5678     \orig@inputref[#1]{#2}  
5679   }  
5680 }
```

(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>}`.

```
5681 \newlength{\slidelogoheight}
5682
5683 \bool_if:NTF \c__notesslides_notes_bool {
5684   \setlength{\slidelogoheight}{.4cm}
5685 }{
5686   \setlength{\slidelogoheight}{1cm}
5687 }
5688 \newsavebox{\slidelogo}
5689 \sbox{\slidelogo}{\TeX}
5690 \newrobustcmd{\setslidelogo}[1]{
5691   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5692 }
```

(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.

```
5693 \def\source{Michael Kohlhase}% customize locally
5694 \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.

```
5695 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5696 \newsavebox{\cclogo}
5697 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5698 \newif\ifcchref\cchreffalse
5699 \AtBeginDocument{
5700   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5701 }
5702 \def\licensing{
5703   \ifcchref
5704     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5705   \else
5706     {\usebox{\cclogo}}
5707   \fi
5708 }
5709 \newrobustcmd{\setlicensing}[2][]{
5710   \def@url{#1}
5711   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5712   \ifx@url@empty
5713     \def\licensing{{\usebox{\cclogo}}}
5714   \else
5715     \def\licensing{
```

```

5716     \ifcchref
5717     \href{#1}{\usebox{\cclogo}}
5718   \else
5719     {\usebox{\cclogo}}
5720   \fi
5721 }
5722 \fi
5723 }

```

(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

```

5724 \newrobustcmd\miko@slidelabel{
5725   \vbox to \slidelogoheight{
5726     \vss\hbox to \slidewidth
5727     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5728   }
5729 }

```

(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.

```

5730 \def\Gin@mhrepos{}
5731 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5732 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5733 \newrobustcmd\frameimage[2][]{
5734   \stepcounter{slide}
5735   \bool_if:NT \c__notesslides_frameimages_bool {
5736     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5737     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5738     \begin{center}
5739       \bool_if:NTF \c__notesslides_fiboxed_bool {
5740         \fbox{
5741           \ifx\Gin@ewidth\@empty
5742             \ifx\Gin@mhrepos\@empty
5743               \mhgraphics[width=\slidewidth,#1]{#2}
5744             \else
5745               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5746             \fi
5747           \else% Gin@ewidth empty
5748             \ifx\Gin@mhrepos\@empty
5749               \mhgraphics[#1]{#2}
5750             \else
5751               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5752             \fi
5753           \fi% Gin@ewidth empty
5754         }
5755       }{
5756         \ifx\Gin@ewidth\@empty

```

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

```

5757         \ifx\Gin@mhrepos\@empty
5758             \mhgraphics[width=\slidewidth,#1]{#2}
5759         \else
5760             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5761         \fi
5762         \ifx\Gin@mhrepos\@empty
5763             \mhgraphics[#1]{#2}
5764         \else
5765             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5766         \fi
5767     \fi% Gin@ewidth empty
5768 }
5769 \end{center}
5770 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5771 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5772 }
5773 } % 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.

```

5774 \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

```

5775 \AddToHook{begindocument}{
5776     \definecolor{green}{rgb}{0,.5,0}
5777     \definecolor{purple}{cmyk}{.3,1,0,.17}
5778 }

```

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`.

```

5779 % \def\STpresent#1{\textcolor{blue}{#1}}
5780 \def\defemph#1{\textcolor{magenta}{#1}}
5781 \def\symrefemph#1{\textcolor{cyan}{#1}}
5782 \def\compemph#1{\textcolor{blue}{#1}}
5783 \def\titleemph#1{\textcolor{blue}{#1}}
5784 \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.

```

5785 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5786 \def\smalltextwarning{
5787     \pgfuseimage{miko@small@dbend}
5788     \xspace
5789 }
5790 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```



```

5791 \newrobustcmd\textwarning{
5792   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5793   \xspace
5794 }
5795 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5796 \newrobustcmd\bigtextwarning{
5797   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5798   \xspace
5799 }

```

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

```

5800 \newrobustcmd\putgraphicsat[3]{
5801   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5802 }
5803 \newrobustcmd\putat[2]{
5804   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5805 }

```

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`.

```

5806 \bool_if:NT \c__notesslides_sectocframes_bool {
5807   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5808     \newcounter{chapter}\counterwithin*{section}{chapter}
5809   }{
5810     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5811       \newcounter{chapter}\counterwithin*{section}{chapter}
5812     }
5813   }
5814 }

```

`\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
5815 \def\part@prefix{}
5816 \@ifpackageloaded{document-structure}{\{
5817   \str_case:VnF \__notesslidesstopsect {
5818     {part}{
5819       \int_set:Nn \l_document_structure_section_level_int {0}
5820       \def\thesection{\arabic{chapter}.\arabic{section}}
5821       \def\part@prefix{\arabic{chapter}.}
5822     }
5823     {chapter}{
5824       \int_set:Nn \l_document_structure_section_level_int {1}
5825       \def\thesection{\arabic{chapter}.\arabic{section}}
5826       \def\part@prefix{\arabic{chapter}.}
5827     }
5828   }{
5829     \int_set:Nn \l_document_structure_section_level_int {2}
5830     \def\part@prefix{}

```

```

5831 }
5832 }
5833
5834 \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

```

5835 \renewenvironment{omgroup}[2][]{
5836   \__document_structure_omgroup_args:n { #1 }
5837   \int_incr:N \l_document_structure_section_level_int
5838   \bool_if:NT \c__notesslides_sectocframes_bool {
5839     \stepcounter{slide}
5840     \begin{frame}[noframenumbering]
5841       \vfill\Large\centering
5842       \red{
5843         \ifcase\l_document_structure_section_level_int\or
5844           \stepcounter{part}
5845           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5846           \def\currentsectionlevel{\omdoc@part@kw}
5847         \or
5848           \stepcounter{chapter}
5849           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5850           \def\currentsectionlevel{\omdoc@chapter@kw}
5851         \or
5852           \stepcounter{section}
5853           \def\__notesslideslabel{\part@prefix\arabic{section}}
5854           \def\currentsectionlevel{\omdoc@section@kw}
5855         \or
5856           \stepcounter{subsection}
5857           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5858           \def\currentsectionlevel{\omdoc@subsection@kw}
5859         \or
5860           \stepcounter{subsubsection}
5861           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5862           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5863         \or
5864           \stepcounter{paragraph}
5865           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
5866           \def\currentsectionlevel{\omdoc@paragraph@kw}
5867         \else
5868           \def\__notesslideslabel{}
5869           \def\currentsectionlevel{\omdoc@paragraph@kw}
5870         \fi% end ifcase
5871         \__notesslideslabel%\sref@label@id\__notesslideslabel
5872         \quad #2%
5873       }%
5874       \vfill%
5875     \end{frame}%
5876   }
5877   \str_if_empty:NF \l__document_structure_omgroup_id_str {
5878     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str

```

```

5879     }
5880   }{}
5881 }

```

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

```

5882 \def\inserttheorembodyfont{\normalfont}
5883 %\bool_if:NF \c__notesslides_notes_bool {
5884 %   \defbeamertemplate{theorem begin}{miko}
5885 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5886 %     \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
5887 %     \inserttheorempunctuation\inserttheorembodyfont\xspace}
5888 %   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5889 % \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.

```

5890 % \expandafter\def\csname Parent2\endcsname{}
5891 %}
5892
5893 \AddToHook{begindocument}{ % this does not work for some reason
5894   \setbeamertemplate{theorems}[ams style]
5895 }
5896 \bool_if:NT \c__notesslides_notes_bool {
5897   \renewenvironment{columns}[1][ ]{%
5898     \par\noindent%
5899     \begin{minipage}%
5900       \slidewidth\centering\leavevmode%
5901   }{%
5902     \end{minipage}\par\noindent%
5903   }%
5904   \newsavebox\columnbox%
5905   \renewenvironment<>{column}[2][ ]{%
5906     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5907   }{%
5908     \end{minipage}\end{lrbox}\usebox\columnbox%
5909   }%
5910 }
5911 \bool_if:NTF \c__notesslides_noproblems_bool {
5912   \newenvironment{problems}{}{}
5913 }{
5914   \excludcomment{problems}
5915 }

```

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.

```

5916 \gdef\printexcursions{}
5917 \newcommand\excursionref[2]{% label, text

```

```

5918 \bool_if:NT \c__notesslides_notes_bool {
5919   \begin{sparagraph}[title=Excursion]
5920     #2 \sref[fallback=the appendix]{#1}.
5921   \end{sparagraph}
5922 }
5923 }
5924 \newcommand\activate@excursion[2][]{
5925   \gappto\printexcursions{\inputref{#1}{#2}}
5926 }
5927 \newcommand\excursion[4][]{% repos, label, path, text
5928   \bool_if:NT \c__notesslides_notes_bool {
5929     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5930   }
5931 }

```

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

\excursiongroup

```

5932 \keys_define:nn{notesslides / excursiongroup }{
5933   id          .str_set_x:N = \l__notesslides_excursion_id_str,
5934   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
5935   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5936 }
5937 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5938   \tl_clear:N \l__notesslides_excursion_intro_tl
5939   \str_clear:N \l__notesslides_excursion_id_str
5940   \str_clear:N \l__notesslides_excursion_mhrepos_str
5941   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5942 }
5943 \newcommand\excursiongroup[1][]{
5944   \__notesslides_excursion_args:n{ #1 }
5945   \ifdefempty\printexcursions{}% only if there are excursions
5946   {\begin{note}
5947     \begin{omgroup}[#1]{Excursions}%
5948     \ifdefempty\l__notesslides_excursion_intro_tl{
5949       \inputref[\l__notesslides_excursion_mhrepos_str]{
5950         \l__notesslides_excursion_intro_tl
5951       }
5952     }
5953     \printexcursions%
5954     \end{omgroup}
5955     \end{note}}
5956 }
5957 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5958 \</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.

```
5959 <*package>
5960 <@@=problems>
5961 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5962 \RequirePackage{l3keys2e,expl-keystr-compatible}
5963
5964 \keys_define:nn { problem / pkg }{
5965   notes      .default:n    = { true },
5966   notes      .bool_set:N   = \c__problems_notes_bool,
5967   gnotes     .default:n    = { true },
5968   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5969   hints      .default:n    = { true },
5970   hints      .bool_set:N   = \c__problems_hints_bool,
5971   solutions  .default:n    = { true },
5972   solutions  .bool_set:N   = \c__problems_solutions_bool,
5973   pts        .default:n    = { true },
5974   pts        .bool_set:N   = \c__problems_pts_bool,
5975   min        .default:n    = { true },
5976   min        .bool_set:N   = \c__problems_min_bool,
5977   boxed      .default:n    = { true },
5978   boxed      .bool_set:N   = \c__problems_boxed_bool,
5979   unknown    .code:n       = {}
5980 }
5981 \newif\ifsolutions
5982
5983 \ProcessKeysOptions{ problem / pkg }
5984 \bool_if:NTF \c__problems_solutions_bool {
5985   \solutionstrue
5986 }{
5987   \solutionsfalse
5988 }
```

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

```
5989 \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.

```
5990 \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.

```
5991 \def\prob@problem@kw{Problem}
5992 \def\prob@solution@kw{Solution}
5993 \def\prob@hint@kw{Hint}
5994 \def\prob@note@kw{Note}
5995 \def\prob@gnote@kw{Grading}
5996 \def\prob@pt@kw{pt}
5997 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5998 \AddToHook{begindocument}{
5999   \ltx@ifpackageloaded{babel}{
6000     \makeatletter
6001     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6002     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6003       \input{problem-ngerman.ldf}
6004     }
6005     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6006       \input{problem-finnish.ldf}
6007     }
6008     \clist_if_in:NnT \l_tmpa_clist {french}{
6009       \input{problem-french.ldf}
6010     }
6011     \clist_if_in:NnT \l_tmpa_clist {russian}{
6012       \input{problem-russian.ldf}
6013     }
6014     \makeatother
6015   }{ }
6016 }
```

40.2 Problems and Solutions

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

```
6017 \keys_define:nn{ problem / problem }{
6018   id      .str_set:x:N = \l__problems_prob_id_str,
6019   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6020   min     .tl_set:N    = \l__problems_prob_min_tl,
6021   title   .tl_set:N    = \l__problems_prob_title_tl,
6022   type    .tl_set:N    = \l__problems_prob_type_tl,
6023   refnum  .int_set:N    = \l__problems_prob_refnum_int
6024 }
6025 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6026 \str_clear:N \l__problems_prob_id_str
6027 \tl_clear:N \l__problems_prob_pts_tl
6028 \tl_clear:N \l__problems_prob_min_tl
6029 \tl_clear:N \l__problems_prob_title_tl
6030 \tl_clear:N \l__problems_prob_type_tl
6031 \int_zero_new:N \l__problems_prob_refnum_int
6032 \keys_set:nn { problem / problem }{ #1 }
6033 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6034   \let\l__problems_prob_refnum_int\undefined
6035 }
6036 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6037 \newcounter{problem}
6038 \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.

```

6039 \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

```

6040 \newcommand\prob@number{
6041   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6042     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6043   }{
6044     \int_if_exist:NTF \l__problems_prob_refnum_int {
6045       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6046     }{
6047       \prob@label\theproblem
6048     }
6049   }
6050 }

```

(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.

```

6051 \newcommand\prob@title[3]{%
6052   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6053     #2 \l__problems_inclprob_title_tl #3
6054   }{
6055     \tl_if_exist:NTF \l__problems_prob_title_tl {
6056       #2 \l__problems_prob_title_tl #3
6057     }{
6058       #1
6059     }
6060   }
6061 }

```

(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.

```

6062 \def\prob@heading{
6063   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6064   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
6065 }

```

(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`

```

6066 \newenvironment{sproblem}[1][{}]{
6067   \__problems_prob_args:n{#1}%\sref@target%
6068   \@in@omtexttrue% we are in a statement (for inline definitions)
6069   \stepcounter{problem}\record@problem
6070   \def\current@section@level{\prob@problem@kw}
6071   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6072     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6073   }{
6074     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6075   }
6076   \str_if_exist:NTF \l__problems_inclprob_id_str {
6077     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6078   }{
6079     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6080   }
6081
6082
6083   \clist_set:No \l_tmpa_clist \sproblemtype
6084   \tl_clear:N \l_tmpa_tl
6085   \clist_map_inline:Nn \l_tmpa_clist {
6086     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6087       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6088     }
6089   }
6090   \tl_if_empty:NTF \l_tmpa_tl {
6091     \__problems_sproblem_start:
6092   }{
6093     \l_tmpa_tl
6094   }
6095   \stex_ref_new_doc_target:n \sproblemid
6096 }{
6097   \clist_set:No \l_tmpa_clist \sproblemtype
6098   \tl_clear:N \l_tmpa_tl
6099   \clist_map_inline:Nn \l_tmpa_clist {
6100     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6101       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6102     }

```



```

6103 }
6104 \tl_if_empty:NTF \l_tmpa_tl {
6105   \__problems_sproblem_end:
6106 }{
6107   \l_tmpa_tl
6108 }
6109
6110
6111 \smallskip
6112 }
6113
6114
6115 \cs_new_protected:Nn \__problems_sproblem_start: {
6116   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6117 }
6118 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6119
6120 \newcommand\stexpatchproblem[3][] {
6121   \str_set:Nx \l_tmpa_str{ #1 }
6122   \str_if_empty:NTF \l_tmpa_str {
6123     \tl_set:Nn \__problems_sproblem_start: { #2 }
6124     \tl_set:Nn \__problems_sproblem_end: { #3 }
6125   }{
6126     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6127     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6128   }
6129 }
6130
6131
6132 \bool_if:NT \c__problems_boxed_bool {
6133   \surroundwithmdframed{problem}
6134 }

```

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

```

6135 \def\record@problem{
6136   \protected@write\@auxout{}
6137   {
6138     \string\@problem{\prob@number}
6139     {
6140       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6141         \l__problems_inclprob_pts_tl
6142       }{
6143         \l__problems_prob_pts_tl
6144       }
6145     }%
6146     {
6147       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6148         \l__problems_inclprob_min_tl
6149       }{
6150         \l__problems_prob_min_tl
6151       }
6152     }
6153   }
6154 }

```

(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).

```
6155 \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.

```
6156 \keys_define:nn { problem / solution }{
6157   id                .str_set:N = \l__problems_solution_id_str ,
6158   for               .tl_set:N  = \l__problems_solution_for_tl ,
6159   height            .dim_set:N = \l__problems_solution_height_dim ,
6160   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6161   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
6162   srccite           .tl_set:N  = \l__problems_solution_srccite_tl
6163 }
6164 \cs_new_protected:Nn \__problems_solution_args:n {
6165   \str_clear:N \l__problems_solution_id_str
6166   \tl_clear:N \l__problems_solution_for_tl
6167   \tl_clear:N \l__problems_solution_srccite_tl
6168   \clist_clear:N \l__problems_solution_creators_clist
6169   \clist_clear:N \l__problems_solution_contributors_clist
6170   \dim_zero:N \l__problems_solution_height_dim
6171   \keys_set:nn { problem / solution }{ #1 }
6172 }
```

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

```
6173 \newcommand\@startsolution[1][{}]{
6174   \__problems_solution_args:n { #1 }
6175   \@in@omtexttrue% we are in a statement.
6176   \bool_if:NF \c__problems_boxed_bool { \hrule }
6177   \smallskip\noindent
6178   {\textbf\prob@solution@kw : \enspace}
6179   \begin{small}
6180   \def\current@section@level{\prob@solution@kw}
6181   \ignorespacesandpars
6182 }
```

\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.

```
6183 \newcommand\startsolutions{
6184   \specialcomment{solution}{\@startsolution}{
6185     \bool_if:NF \c__problems_boxed_bool {
6186       \hrule\medskip
6187     }
6188     \end{small}%
6189   }
6190   \bool_if:NT \c__problems_boxed_bool {
6191     \surroundwithmdframed{solution}
6192   }
6193 }
```

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

\stopsolutions

```
6194 \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.

```
6195 \ifsolutions
6196   \startsolutions
6197 \else
6198   \stopsolutions
6199 \fi
```

exnote

```
6200 \bool_if:NTF \c__problems_notes_bool {
6201   \newenvironment{exnote}[1][]{
6202     \par\smallskip\hrule\smallskip
6203     \noindent\textbf{\prob@note@kw : }\small
6204   }{
6205     \smallskip\hrule
6206   }
6207 }{
6208   \excludecomment{exnote}
6209 }
```

hint

```
6210 \bool_if:NTF \c__problems_notes_bool {
6211   \newenvironment{hint}[1][]{
6212     \par\smallskip\hrule\smallskip
6213     \noindent\textbf{\prob@hint@kw :~ }\small
6214   }{
6215     \smallskip\hrule
6216   }
6217   \newenvironment{exhint}[1][]{
6218     \par\smallskip\hrule\smallskip
6219     \noindent\textbf{\prob@hint@kw :~ }\small
6220   }{
6221     \smallskip\hrule
6222   }
6223 }{
6224   \excludecomment{hint}
6225   \excludecomment{exhint}
6226 }
```

gnote

```
6227 \bool_if:NTF \c__problems_notes_bool {
6228   \newenvironment{gnote}[1][]{
6229     \par\smallskip\hrule\smallskip
6230     \noindent\textbf{\prob@gnote@kw : }\small
6231   }{
6232     \smallskip\hrule
6233   }
6234 }{
6235   \excludecomment{gnote}
6236 }
```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6237 \newenvironment{mcb}{
6238   \begin{enumerate}
6239 }{
6240   \end{enumerate}
6241 }
```

we define the keys for the mcc macro

```
6242 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6243   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6244     \bool_set_true:N #1
6245   }{
6246     \bool_set_false:N #1
6247   }
6248 }
6249 \keys_define:nn { problem / mcc }{
6250   id          .str_set:x:N = \l__problems_mcc_id_str ,
6251   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6252   T           .default:n   = { true } ,
6253   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6254   F           .default:n   = { true } ,
6255   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6256   Ttext       .code:n      = {
6257     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6258   } ,
6259   Ftext       .code:n      = {
6260     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6261   }
6262 }
6263 \cs_new_protected:Nn \l__problems_mcc_args:n {
6264   \str_clear:N \l__problems_mcc_id_str
6265   \tl_clear:N \l__problems_mcc_feedback_tl
6266   \bool_set_true:N \l__problems_mcc_t_bool
6267   \bool_set_true:N \l__problems_mcc_f_bool
6268   \bool_set_true:N \l__problems_mcc_Ttext_bool
6269   \bool_set_false:N \l__problems_mcc_Ftext_bool
6270   \keys_set:nn { problem / mcc }{ #1 }
6271 }
```

\mcc

```
6272 \newcommand\mcc[2][] {
6273   \l__problems_mcc_args:n{ #1 }
6274   \item #2
6275   \ifsolutions
6276     \
6277     \bool_if:NT \l__problems_mcc_t_bool {
6278       % TODO!
6279       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
6280     }
6281     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

6282      % TODO!
6283      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6284    }
6285    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6286      !
6287    }{
6288      \l__problems_mcc_feedback_tl
6289    }
6290    \fi
6291  } %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.

```

6292
6293 \keys_define:nn{ problem / inclproblem }{
6294   id      .str_set_x:N = \l__problems_inclprob_id_str,
6295   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
6296   min     .tl_set:N    = \l__problems_inclprob_min_tl,
6297   title   .tl_set:N    = \l__problems_inclprob_title_tl,
6298   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
6299   type    .tl_set:N    = \l__problems_inclprob_type_tl,
6300   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
6301 }
6302 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6303   \str_clear:N \l__problems_prob_id_str
6304   \tl_clear:N \l__problems_inclprob_pts_tl
6305   \tl_clear:N \l__problems_inclprob_min_tl
6306   \tl_clear:N \l__problems_inclprob_title_tl
6307   \tl_clear:N \l__problems_inclprob_type_tl
6308   \int_zero_new:N \l__problems_inclprob_refnum_int
6309   \str_clear:N \l__problems_inclprob_mhrepos_str
6310   \keys_set:nn { problem / inclproblem }{ #1 }
6311   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6312     \let\l__problems_inclprob_pts_tl\undefined
6313   }
6314   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6315     \let\l__problems_inclprob_min_tl\undefined
6316   }
6317   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6318     \let\l__problems_inclprob_title_tl\undefined
6319   }
6320   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6321     \let\l__problems_inclprob_type_tl\undefined
6322   }
6323   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6324     \let\l__problems_inclprob_refnum_int\undefined
6325   }
6326 }

```

```

6327
6328 \cs_new_protected:Nn \__problems_inclprob_clear: {
6329   \let\l__problems_inclprob_id_str\undefined
6330   \let\l__problems_inclprob_pts_tl\undefined
6331   \let\l__problems_inclprob_min_tl\undefined
6332   \let\l__problems_inclprob_title_tl\undefined
6333   \let\l__problems_inclprob_type_tl\undefined
6334   \let\l__problems_inclprob_refnum_int\undefined
6335   \let\l__problems_inclprob_mhrepos_str\undefined
6336 }
6337 \__problems_inclprob_clear:
6338
6339 \newcommand\includeproblem[2][ ]{
6340   \__problems_inclprob_args:n{ #1 }
6341   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6342     \input{#2}
6343   }{
6344     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6345       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6346     }
6347   }
6348   \__problems_inclprob_clear:
6349 }

```

(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.

```

6350 \AddToHook{enddocument}{
6351   \bool_if:NT \c__problems_pts_bool {
6352     \message{Total:~\arabic{pts}~points}
6353   }
6354   \bool_if:NT \c__problems_min_bool {
6355     \message{Total:~\arabic{min}~minutes}
6356   }
6357 }

```

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

```

6358 \def\pts#1{
6359   \bool_if:NT \c__problems_pts_bool {
6360     \marginpar{#1~\prob@pt@kw}
6361   }
6362 }
6363 \def\min#1{
6364   \bool_if:NT \c__problems_min_bool {
6365     \marginpar{#1~\prob@min@kw}
6366   }
6367 }

```

`\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.

```

6368 \newcounter{pts}
6369 \def\show@pts{
6370   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6371     \bool_if:NT \c__problems_pts_bool {
6372       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6373       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6374     }
6375   }{
6376     \tl_if_exist:NT \l__problems_prob_pts_tl {
6377       \bool_if:NT \c__problems_pts_bool {
6378         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6379         \addtocounter{pts}{\l__problems_prob_pts_tl}
6380       }
6381     }
6382   }
6383 }

```

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

`\show@min`

```

6384 \newcounter{min}
6385 \def\show@min{
6386   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6387     \bool_if:NT \c__problems_min_bool {
6388       \marginpar{\l__problems_inclprob_min_tl\ min}
6389       \addtocounter{min}{\l__problems_inclprob_min_tl}
6390     }
6391   }{
6392     \tl_if_exist:NT \l__problems_prob_min_tl {
6393       \bool_if:NT \c__problems_min_bool {
6394         \marginpar{\l__problems_prob_min_tl\ min}
6395         \addtocounter{min}{\l__problems_prob_min_tl}
6396       }
6397     }
6398   }
6399 }
6400 </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.

```
6401 <@@=hwexam>
6402 <*cls>
6403 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6404 \RequirePackage{l3keys2e,expl-keystr-compat}
6405 \DeclareOption*{
6406   \PassOptionsToClass{\CurrentOption}{document-structure}
6407   \PassOptionsToPackage{\CurrentOption}{stex}
6408   \PassOptionsToPackage{\CurrentOption}{hwexam}
6409   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6410 }
6411 \ProcessOptions
```

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

```
6412 \LoadClass{document-structure}
6413 \RequirePackage{stex}
6414 \RequirePackage{hwexam}
6415 \RequirePackage{tikzinput}
6416 \RequirePackage{graphicx}
6417 \RequirePackage{a4wide}
6418 \RequirePackage{amssymb}
6419 \RequirePackage{amstext}
6420 \RequirePackage{amsmath}
```

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


```

6421 \newcommand\assig@default@type{\hwexam@assignment@kw}
6422 \def\document@hwexamtype{\assig@default@type}
6423 <@@=document_structure>
6424 \keys_define:nn { document-structure / document }{
6425   id .str_set_x:N = \c_document_structure_document_id_str,
6426   hwexamtype .tl_set:N = \document@hwexamtype
6427 }
6428 <@@=hwexam>
6429 </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.

```
6430 \*package>
6431 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6432 \RequirePackage{l3keys2e,expl-keystr-compat}
6433
6434 \newif\iftest\testfalse
6435 \DeclareOption{test}{\testtrue}
6436 \newif\ifmultiple\multiplefalse
6437 \DeclareOption{multiple}{\multipletrue}
6438 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6439 \ProcessOptions
```

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

```
6440 \RequirePackage{keyval}[1997/11/10]
6441 \RequirePackage{problem}
```

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

```
6442 \newcommand\hwexam@assignment@kw{Assignment}
6443 \newcommand\hwexam@given@kw{Given}
6444 \newcommand\hwexam@due@kw{Due}
6445 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6446 blank~for~extra~space}
6447 \def\hwexam@minutes@kw{minutes}
6448 \newcommand\correction@probs@kw{prob.}
6449 \newcommand\correction@pts@kw{total}
6450 \newcommand\correction@reached@kw{reached}
6451 \newcommand\correction@sum@kw{Sum}
6452 \newcommand\correction@grade@kw{grade}
6453 \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

```

6454 \AddToHook{begindocument}{
6455 \ltx@ifpackageloaded{babel}{
6456 \makeatletter
6457 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6458 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6459 \input{hwexam-ngerman.ldf}
6460 }
6461 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6462 \input{hwexam-finnish.ldf}
6463 }
6464 \clist_if_in:NnT \l_tmpa_clist {french}{
6465 \input{hwexam-french.ldf}
6466 }
6467 \clist_if_in:NnT \l_tmpa_clist {russian}{
6468 \input{hwexam-russian.ldf}
6469 }
6470 \makeatother
6471 }{}
6472 }
6473

```

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.

```

6474 \newcounter{assignment}
6475 \numberproblemsin{assignment}
6476 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6477 \keys_define:nn { hwexam / assignment } {
6478 id .str_set:N = \l__hwexam_assign_id_str,
6479 number .int_set:N = \l__hwexam_assign_number_int,
6480 title .tl_set:N = \l__hwexam_assign_title_tl,
6481 type .tl_set:N = \l__hwexam_assign_type_tl,
6482 given .tl_set:N = \l__hwexam_assign_given_tl,
6483 due .tl_set:N = \l__hwexam_assign_due_tl,
6484 loadmodules .code:n = {
6485 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6486 }
6487 }
6488 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6489 \str_clear:N \l__hwexam_assign_id_str
6490 \int_set:Nn \l__hwexam_assign_number_int {-1}
6491 \tl_clear:N \l__hwexam_assign_title_tl
6492 \tl_clear:N \l__hwexam_assign_type_tl
6493 \tl_clear:N \l__hwexam_assign_given_tl
6494 \tl_clear:N \l__hwexam_assign_due_tl
6495 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6496 \keys_set:nn { hwexam / assignment }{ #1 }
6497 }

```

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.

```

6498 \newcommand\given@due[2]{
6499 \bool_lazy_all:nF {
6500 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6501 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6502 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6503 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6504 }{ #1 }
6505
6506 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6507 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6508 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6509 }
6510 }{
6511 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6512 }
6513
6514 \bool_lazy_or:nnF {
6515 \bool_lazy_and_p:nn {
6516 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6517 }{
6518 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6519 }
6520 }{
6521 \bool_lazy_and_p:nn {
6522 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6523 }{
6524 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6525 }
6526 }{ ,~ }
6527
6528 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6529 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6530 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6531 }
6532 }{
6533 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6534 }
6535
6536 \bool_lazy_all:nF {
6537 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6538 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6539 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6540 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6541 }{ #2 }
6542 }

```

`\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.

```

6543 \newcommand\assignment@title[3]{
6544 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6545 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6546 #1
6547 }{
6548 #2\l__hwexam_assign_title_tl#3
6549 }
6550 }{
6551 #2\l__hwexam_inclasssign_title_tl#3
6552 }
6553 }

```

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

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

```

6554 \newcommand\assignment@number{
6555 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6556 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6557 \arabic{assignment}
6558 } {
6559 \int_use:N \l__hwexam_assign_number_int
6560 }
6561 }{
6562 \int_use:N \l__hwexam_inclasssign_number_int
6563 }
6564 }

```

(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.

```

6565 \newenvironment{assignment}[1][ ]{
6566 \__hwexam_assignment_args:n { #1 }
6567 %\sref@target
6568 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6569 \global\stepcounter{assignment}
6570 }{
6571 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6572 }
6573 \setcounter{problem}{0}
6574 \def\current@section@level{\document@hwexamtype}
6575 %\sref@label@id{\document@hwexamtype \thesection}
6576 \begin{@assignment}
6577 }{
6578 \end{@assignment}
6579 }

```

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

```

6580 \def\ass@title{
6581 \protect\document@hwexamtype~\arabic{assignment}
6582 \assignment@title{}\{;\}{} -- \given@due{}\}{}
6583 }
6584 \ifmultiple
6585 \newenvironment{@assignment}{
6586 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6587 \begin{omgroup}[loadmodules]{\ass@title}
6588 }{
6589 \begin{omgroup}{\ass@title}
6590 }
6591 }{
6592 \end{omgroup}
6593 }

```

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

```

6594 \else
6595 \newenvironment{@assignment}{
6596 \begin{center}\bf
6597 \Large@title\strut\\
6598 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
6599 \large\given@due{--;\}{}{;\}{}
6600 \end{center}
6601 }{}
6602 \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.

```

6603 \keys_define:nn { hwexam / inclassignment } {
6604 %id .str_set_x:N = \l__hwexam_assign_id_str,
6605 number .int_set:N = \l__hwexam_inclassign_number_int,
6606 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6607 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6608 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6609 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6610 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6611 }
6612 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6613 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6614 \tl_clear:N \l__hwexam_inclassign_title_tl
6615 \tl_clear:N \l__hwexam_inclassign_type_tl
6616 \tl_clear:N \l__hwexam_inclassign_given_tl
6617 \tl_clear:N \l__hwexam_inclassign_due_tl
6618 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6619 \keys_set:nn { hwexam / inclassignment }{ #1 }
6620 }
6621 \__hwexam_inclassignment_args:n {}
6622
6623 \newcommand\inputassignment[2][{}]{

```

```

6624 \_hwexam_inclassnment_args:n { #1 }
6625 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6626 \input{#2}
6627 }{
6628 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6629 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6630 }
6631 }
6632 \_hwexam_inclassnment_args:n {}
6633 }
6634 \newcommand\includeassignment[2][ ]{
6635 \newpage
6636 \inputassignment[#1]{#2}
6637 }

```

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

42.4 Typesetting Exams

\quizheading

```

6638 \ExplSyntaxOff
6639 \newcommand\quizheading[1]{%
6640 \def\@tas{#1}%
6641 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6642 \ifx\@tas\@empty\else%
6643 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6644 \fi%
6645 }
6646 \ExplSyntaxOn

```

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

\testheading

```

6647
6648 \def\hwexamheader{\input{hwexam-default.header}}
6649
6650 \def\hwexamminutes{
6651 \tl_if_empty:NTF \testheading@duration {
6652 {\testheading@min}~\hwexam@minutes@kw
6653 }{
6654 \testheading@duration
6655 }
6656 }
6657
6658 \keys_define:nn { hwexam / testheading } {
6659 min .tl_set:N = \testheading@min,
6660 duration .tl_set:N = \testheading@duration,
6661 reqpts .tl_set:N = \testheading@reqpts,
6662 tools .tl_set:N = \testheading@tools
6663 }
6664 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6665 \tl_clear:N \testheading@min
6666 \tl_clear:N \testheading@duration

```

```

6667 \tl_clear:N \testheading@reqpts
6668 \tl_clear:N \testheading@tools
6669 \keys_set:nn { hwexam / testheading }{ #1 }
6670 }
6671 \newenvironment{testheading}[1][]{
6672   \_hwexam_testheading_args:n{ #1 }
6673   \newcount\check@time\check@time=\testheading@min
6674   \advance\check@time by -\theassignment@totalmin
6675   \newif\if@bonuspoints
6676   \tl_if_empty:NTF \testheading@reqpts {
6677     \@bonuspointsfalse
6678   }{
6679     \newcount\bonus@pts
6680     \bonus@pts=\theassignment@totalpts
6681     \advance\bonus@pts by -\testheading@reqpts
6682     \edef\bonus@pts{\the\bonus@pts}
6683     \@bonuspointstrue
6684   }
6685   \edef\check@time{\the\check@time}
6686
6687   \makeatletter\hwexamheader\makeatother
6688 }{
6689   \newpage
6690 }

```

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

\testspace

```

6691 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6692 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6693 \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.

```

6694 <@=problems>
6695 \renewcommand\@problem[3]{
6696   \stepcounter{assignment@probs}
6697   \def\__problemspts{#2}
6698   \ifx\__problemspts\@empty\else
6699     \addtocounter{assignment@totalpts}{#2}
6700   \fi
6701   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6702   \xdef\correction@probs{\correction@probs & #1}%
6703   \xdef\correction@pts{\correction@pts & #2}
6704   \xdef\correction@reached{\correction@reached &}

```



```

6705 }
6706 <@@=hwexam>

```

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

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

```

6707 \newcounter{assignment@probs}
6708 \newcounter{assignment@totalpts}
6709 \newcounter{assignment@totalmin}
6710 \def\correction@probs{\correction@probs@kw}
6711 \def\correction@pts{\correction@pts@kw}
6712 \def\correction@reached{\correction@reached@kw}
6713 \stepcounter{assignment@probs}
6714 \newcommand\correction@table{
6715 \resizebox{\textwidth}{!}{%
6716 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6717 &\multicolumn{\theassignment@probs}{c|}|%|
6718 {\footnotesize\correction@forgrading@kw} &\\ \hline
6719 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6720 \correction@pts & \theassignment@totalpts & \\ \hline
6721 \correction@reached & & \[.7cm]\hline
6722 \end{tabular}}
6723 </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}

```