

The sTeX3 Package *

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

2022-02-16

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 2022-02-16)

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	170
33.4	Logical Paragraphs	173

34 The Implementation	177
34.1 Package Options	177
34.2 Proofs	177
34.3 Justifications	183
35 \LaTeX-Others Implementation	185
36 \LaTeX-Metatheory Implementation	186
37 Tikzinput Implementation	189
38 document-structure.sty Implementation	191
38.1 The document-structure Class	191
38.2 Class Options	191
38.3 Beefing up the <code>document</code> environment	192
38.4 Implementation: document-structure Package	192
38.5 Package Options	192
38.6 Document Structure	194
38.7 Front and Backmatter	197
38.8 Global Variables	199
39 NotesSlides – Implementation	200
39.1 Class and Package Options	200
39.2 Notes and Slides	202
39.3 Header and Footer Lines	206
39.4 Frame Images	207
39.5 Colors and Highlighting	208
39.6 Sectioning	209
39.7 Excursions	211
40 The Implementation	213
40.1 Package Options	213
40.2 Problems and Solutions	214
40.3 Multiple Choice Blocks	220
40.4 Including Problems	221
40.5 Reporting Metadata	222
41 Implementation: The hwexam Class	224
41.1 Class Options	224
42 Implementation: The hwexam Package	226
42.1 Package Options	226
42.2 Assignments	227
42.3 Including Assignments	230
42.4 Typesetting Exams	231
42.5 Leftovers	233

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/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

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

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

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

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

Module 2:
Module 3:
Module 4:

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

Example 3

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

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

Module 7: For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 5

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

ab

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

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

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

Example 6

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

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

.

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

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

Example 7

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

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

.

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

Example 8

```
\mult[\comp{Multiplying}]* $a$  $b$  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  $P$  holds for every  $x \in A$ }
```

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

⁴EdNOTE: TODO

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

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

Example 10

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

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

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

Example 11

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

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

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

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

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

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

There's also a convenient 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}
 $\text{\textcolor{teal}{$}\textcolor{teal}{plus}\textcolor{teal}{\{a\}}\textcolor{teal}{\{\textcolor{teal}{times}\textcolor{teal}{\{b\}}\textcolor{teal}{\{c\}}\}}\textcolor{teal}{\$}}$  and  $\textcolor{teal}{\$}\textcolor{teal}{times}\textcolor{teal}{\{a\}}\textcolor{teal}{\{\textcolor{teal}{plus}\textcolor{teal}{\{b\}}\textcolor{teal}{\{c\}}\}}\textcolor{teal}{\$}$ 
```

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

8.1.2 Archives and Imports

Namespaces

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

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

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 9

sTeX-Basics

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

debug ($\langle log-prefix \rangle *$) 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

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

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

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

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

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

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

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

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

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

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

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

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

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

Chapter 10

sTEX-MathHub

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

10.1 Macros and Environments

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

10.1.1 Files, Paths, URIs

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

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

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

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

<code>\stex_path_if_absolute_p:N</code>	\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 .sms-file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

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

```

```

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

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

Namespace 1:
file://stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

12.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

Module 10:  Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

Test 5

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

Module 11: FooBar Module path: <http://mathhub.info/tests/Foo/Bar/Foo?Bar>
Language:
Signature:
Metatheory:

\STEXModule \STEXModule {<fragment>}

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

\stex_invoke_module:n

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

Test 6

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

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

`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

```
Module 15:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
              Meaning: >macro:->\protect \bar <
Module 16:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 17:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

`\usemodule`

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

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

Test 9

```

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

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,-} \\
All-symbols:-
\seq_use:Nn \l_stex_all_symbols_seq {,-}
\ExplSyntaxOff
\end{smodule}

```

Module 18:

Module 19: Meaning: »macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}<

Module 20: Meaning: »undefined<

Meaning: »macro:->\stex_invoke_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory, file://stextest?UseTest3, file://stextest?UseTest2>

All symbols: <http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?module-type, 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?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?module-type, 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

$\backslash\mathrm{symdecl}$	$\backslash\mathrm{symdecl}[\langle\mathit{args}\rangle]\{\langle\mathit{macroname}\rangle\}$
------------------------------	---

Declares a new symbol with semantic macro $\backslash\mathit{macroname}$. Optional arguments are:

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 23:

`\symdef`

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

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

Test 13

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

Module 24: $a + b + c$

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

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

`\withbrackets`

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

Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$.

Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after `\left` and `\right` in display-mode.

Test 14

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

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

Test 15

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

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{##1 \comp+ ##2}
\notation[prec=100]{ mult }{#1}{##1 \comp\cdot ##2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$
$[\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: $\langle a \mid [b;c;d;e;f]^g \rangle$ and $\langle a \mid [b;c]^g \rangle$ and $\langle a \mid [b]^c \rangle$

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

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

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

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{smodule}
```

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

`\stex_highlight_term:nn`

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

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

`\comp`
`\compemph`
`\compemph@uri`
`\defemph`
`\defemph@uri`
`\symrefemph`
`\symrefemph@uri`

`\comp{<args>}`

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

TeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

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

P.1.1 $n > 1$:

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

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

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

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

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

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

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

18.2.4 Proof Structure

subproof	The pfcases environment is used to mark up a subproof. This environment takes an optional KeyVal argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the proof environment). The method key can be used to give the name of the proof method executed to make this subproof.
method	
spfcases	The pfcases environment is used to mark up a proof by cases. Technically it is a variant of the subproof where the method is by-cases . Its contents are 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 to, or what we have achieved so far. As such, it cannot be the target of a \premise .

18.2.5 Proof End Markers

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

`\sproofend`

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

`\sProofEndSymbol`

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

18.2.6 Configuration of the Presentation

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

EdN:8

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

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

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

The `document-structure` package is part of the \S T E X collection, a version of T E X/L A T E X that allows to markup T E X/L A T E X documents semantically without leaving the document format, essentially turning T E X/L A T E X into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in L A T E X . This includes a simple structure sharing mechanism for \S T E X 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 T E X sources, or after translation.

21.1 Introduction

\S T E X is a version of T E X/L A T E X that allows to markup T E X/L A T E X documents semantically without leaving the document format, essentially turning T E X/L A T E X 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 T E X sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the \S T E X collection.

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

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

21.2 The User Interface

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

21.2.1 Package and Class Options

The `document-structure` class accept the following options:

<code>class=<name></code>	load <code><name>.cls</code> instead of <code>article.cls</code>
<code>topsect=<sect></code>	The top-level sectioning level; the default for <code><sect></code> is <code>section</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any \TeX packages

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

21.2.2 Document Structure

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

```

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

```

⁹EdNOTE: integrate with latexml's XMRef in the Math mode.

²We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindomgroup`

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a `subsection*` as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

`ignore` The `ignore` environment can be used for hiding text parts from the document structure.
`showignores` The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `document-structure` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

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

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

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

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

21.2.4 Structure Sharing

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

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

`\STRsemantics`

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

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

22.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

22.2.3 Header and Footer Lines of the Slides

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

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

22.2.4 Frame Images

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

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV
Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compat}
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_EXML, R_US_TE_X, p_DF_LA_TE_X).

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

```

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

```

```

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

```

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

25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

259 \clist_if_empty:NF \c_stex_languages_clist {
260   \clist_clear:N \l_tmpa_clist
261   \clist_map_inline:Nn \c_stex_languages_clist {
262     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
263       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
264     } {
265       \msg_error:nnx{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 }
```

```

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 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
909   \str_clear:N \l__stex_refs_curr_label_str
910   \str_set:Nx \l_tmpa_str { #1 }
911   \str_if_empty:NF \l_tmpa_str {
912     \stex_get_document_uri:
913     \str_set:Nx \l__stex_refs_curr_label_str {
914       \l_stex_current_docns_str?#1
915     }
916     \seq_if_exist:cF{g__stex_refs_labels_#1_seq}{
917       \seq_new:c {g__stex_refs_labels_#1_seq}
918     }
919     \seq_if_in:coF{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str {
920       \seq_gput_right:co{g__stex_refs_labels_#1_seq}\l__stex_refs_curr_label_str
921     }
922     \stex_if_smsmode:TF {
923       \stex_get_document_url:
924       \str_gset_eq:cN {sref_url_}\l__stex_refs_curr_label_str _str}\l_stex_current_docurl_str
925       \str_gset_eq:cN {sref_}\l__stex_refs_curr_label_str _type}\c__stex_refs_url_str
926     }{

```

```

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

```

27.3 Using References

```

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

```

```

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

```

```

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

```

```
1086     }  
1087   }  
1088  
1089 </package>
```


Chapter 28

STEX -Modules Implementation

```
1090 <*package>
1091
1092 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1093
1094 <@@=stex_modules>
1095
1096   Warnings and error messages
1097 \msg_new:nnn{stex}{error/unknownmodule}{
1098   No~module~#1~found
1099 }
1100 \msg_new:nnn{stex}{error/syntax}{
1101   Syntax~error:~#1
1102 }
1103 \msg_new:nnn{stex}{error/siglanguage}{
1104   Module~#1~declares~signature~#2,~but~does~not~
1105   declare~its~language
1106 }
1107 \msg_new:nnn{stex}{warning/deprecated}{
1108   #1~is~deprecated;~please~use~#2~instead!
1109 }
1110 \msg_new:nnn{stex}{error/conflictingmodules}{
1111   Conflicting~imports~for~module~#1
1112 }
1113
1114 \l_stex_current_module_str The current module:
1115 \str_new:N \l_stex_current_module_str
1116
1117 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1118
1119 \l_stex_all_modules_seq Stores all available modules
1120 \seq_new:N \l_stex_all_modules_seq
1121
1122 (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
1114 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1115   \str_if_empty:NTF \l_stex_current_module_str
1116   \prg_return_false: \prg_return_true:
1117 }

```

(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
1118 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1119   \prop_if_exist:cTF { c_stex_module_#1_prop }
1120   \prg_return_true: \prg_return_false:
1121 }

```

(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
1122 \cs_new_protected:Nn \stex_add_to_current_module:n {
1123   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1124 }
1125 \cs_new_protected:Npn \STEXexport {
1126   \beginngroup
1127   \newlinechar=-1\relax
1128   \endlinechar=-1\relax
1129   %\catcode'\ = 9\relax
1130   \expandafter\endgroup\STEXexport:n
1131 }
1132 \cs_new_protected:Nn \STEXexport:n {
1133   \ignorespaces #1
1134   \stex_add_to_current_module:n { \ignorespaces #1 }
1135   \stex_smsmode_do:
1136 }
1137 \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
1138 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1139   \str_set:Nx \l_tmpa_str { #1 }
1140   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1141 }
1142
1143 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1144 % \str_set:Nx \l_tmpa_str { #1 }
1145 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1146 %}

```

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

```

\stex_collect_imports:n
1147 \cs_new_protected:Nn \stex_collect_imports:n {
1148   \seq_clear:N \l_stex_collect_imports_seq
1149   \__stex_modules_collect_imports:n {#1}

```

```

1150 }
1151 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1152   \seq_map_inline:cn {c_stex_module_#1_imports} {
1153     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1154       \__stex_modules_collect_imports:n { ##1 }
1155     }
1156   }
1157   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1158     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1159   }
1160 }

```

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

`\stex_add_import_to_current_module:n`

```

1161 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1162   \str_set:Nx \l_tmpa_str { #1 }
1163   \exp_args:Nno
1164   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1165     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1166   }
1167 }

```

(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).

```

1168 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1169   \str_set:Nx \l_tmpa_str { #1 }
1170   \seq_set_eq:NN \l_tmpa_seq #2
1171   % split off file extension
1172   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1173   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1174   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1175   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1176
1177   \bool_set_true:N \l_tmpa_bool
1178   \bool_while_do:Nn \l_tmpa_bool {
1179     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1180     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1181       {source} { \bool_set_false:N \l_tmpa_bool }
1182     }{}{
1183       \seq_if_empty:NT \l_tmpa_seq {
1184         \bool_set_false:N \l_tmpa_bool
1185       }
1186     }
1187   }
1188
1189   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1190   \str_if_empty:NTF \l_stex_modules_subpath_str {
1191     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1192   }{
1193     \str_set:Nx \l_stex_modules_ns_str {
1194       \l_tmpa_str/\l_stex_modules_subpath_str

```

```

1195     }
1196   }
1197 }

```

(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
1198 \str_new:N \l_stex_modules_ns_str
1199 \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.

```

1200 \cs_new_protected:Nn \stex_modules_current_namespace: {
1201   \str_clear:N \l_stex_modules_subpath_str
1202   \prop_if_exist:NTF \l_stex_current_repository_prop {
1203     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1204     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1205   }{
1206     % split off file extension
1207     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1208     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1209     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1210     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1211     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1212     \str_set:Nx \l_stex_modules_ns_str {
1213       file:/\stex_path_to_string:N \l_tmpa_seq
1214     }
1215   }
1216 }

```

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

28.1 The module environment

module arguments:

```

1217 \keys_define:nn { stex / module } {
1218   title      .tl_set:N      = \smodulename ,
1219   type       .str_set_x:N   = \smodulename ,
1220   id         .str_set_x:N   = \smodulename ,
1221   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1222   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1223   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1224   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1225   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1226   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1227   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1228   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1229 }
1230
1231 \cs_new_protected:Nn \__stex_modules_args:n {

```

```

1232 \str_clear:N \smoduletitle
1233 \str_clear:N \smoduletype
1234 \str_clear:N \smoduleid
1235 \str_clear:N \l_stex_module_ns_str
1236 \str_clear:N \l_stex_module_deprecate_str
1237 \str_clear:N \l_stex_module_lang_str
1238 \str_clear:N \l_stex_module_sig_str
1239 \str_clear:N \l_stex_module_creators_str
1240 \str_clear:N \l_stex_module_contributors_str
1241 \str_clear:N \l_stex_module_meta_str
1242 \str_clear:N \l_stex_module_srccite_str
1243 \keys_set:nn { stex / module } { #1 }
1244 }
1245
1246 % module parameters here? In the body?
1247

```

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

```

1248 \cs_new_protected:Nn \stex_module_setup:nn {
1249   \str_set:Nx \l_stex_module_name_str { #2 }
1250   \__stex_modules_args:n { #1 }
1251
1252   First, we set up the name and namespace of the module.
1253   Are we in a nested module?
1254
1255   \stex_if_in_module:TF {
1256     % Nested module
1257     \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1258       { ns } \l_stex_module_ns_str
1259     \str_set:Nx \l_stex_module_name_str {
1260       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1261         { name } / \l_stex_module_name_str
1262     }
1263   }{
1264     % not nested:
1265     \str_if_empty:NT \l_stex_module_ns_str {
1266       \stex_modules_current_namespace:
1267       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1268       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1269         / {\l_stex_module_ns_str}
1270       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1271       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1272         \str_set:Nx \l_stex_module_ns_str {
1273           \stex_path_to_string:N \l_tmpa_seq
1274         }
1275       }
1276     }
1277   }
1278 }

```

Next, we determine the language of the module:

```

1274 \str_if_empty:NT \l_stex_module_lang_str {
1275   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1276   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1277   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1278   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>

```

```

1279 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1280 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1281   inferred~from~file~name}
1282 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1283 }
1284 }
1285
1286 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1287   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1288   \l_tmpa_str {
1289     \ltx@ifpackageloaded{babel}{
1290       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1291     }{}
1292   } {
1293     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1294   }
1295 }}

```

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

```

1296 \str_if_empty:NNTF \l_stex_module_sig_str {
1297   \exp_args:Nnx \prop_gset_from_keyval:cn {
1298     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1299   } {
1300     name      = \l_stex_module_name_str ,
1301     ns        = \l_stex_module_ns_str ,
1302     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1303     lang      = \l_stex_module_lang_str ,
1304     sig       = \l_stex_module_sig_str ,
1305     deprecate = \l_stex_module_deprecate_str ,
1306     meta      = \l_stex_module_meta_str
1307   }
1308   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1309   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1310   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1311   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1312   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1313 \str_if_empty:NT \l_stex_module_meta_str {
1314   \str_set:Nx \l_stex_module_meta_str {
1315     \c_stex_metatheory_ns_str ? Metatheory
1316   }
1317 }
1318 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1319   \bool_set_true:N \l_stex_in_meta_bool
1320   \exp_args:Nx \stex_add_to_current_module:n {
1321     \bool_set_true:N \l_stex_in_meta_bool
1322     \stex_activate_module:n {\l_stex_module_meta_str}
1323     \bool_set_false:N \l_stex_in_meta_bool
1324   }
1325   \stex_activate_module:n {\l_stex_module_meta_str}
1326   \bool_set_false:N \l_stex_in_meta_bool
1327 }

```

```

1328 }{
1329   \str_if_empty:NT \l_stex_module_lang_str {
1330     \msg_error:nnxx{stex}{error/siglanguage}{
1331       \l_stex_module_ns_str?\l_stex_module_name_str
1332     }\l_stex_module_sig_str}
1333   }
1334
1335   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1336   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1337   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1338   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1339   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1340   \str_set:Nx \l_tmpa_str {
1341     \stex_path_to_string:N \l_tmpa_seq /
1342     \l_tmpa_str . \l_stex_module_sig_str .tex
1343   }
1344   \IfFileExists \l_tmpa_str {
1345     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1346       \str_clear:N \l_stex_current_module_str
1347       \seq_clear:N \l_stex_all_modules_seq
1348       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1349     }
1350   }{
1351     \msg_error:nnxx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1352   }
1353   \stex_if_smsmode:F {
1354     \stex_activate_module:n {
1355       \l_stex_module_ns_str ? \l_stex_module_name_str
1356     }
1357   }
1358   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1359 }
1360 \str_if_empty:NF \l_stex_module_deprecate_str {
1361   \msg_warning:nnxx{stex}{warning/deprecated}{
1362     Module~\l_stex_current_module_str
1363   }{
1364     \l_stex_module_deprecate_str
1365   }
1366 }
1367 }

```

(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}
1368 \int_new:N \l_stex_module_group_depth_int
1369 \cs_new_protected:Nn \__stex_modules_begin_module: {
1370   \stex_reactivate_macro:N \STEXexport
1371   \stex_reactivate_macro:N \importmodule
1372   \stex_reactivate_macro:N \symdecl
1373   \stex_reactivate_macro:N \notation
1374   \stex_reactivate_macro:N \symdef
1375 }

```

```

1376 \stex_debug:nn{modules}{
1377   New~module:\\
1378   Namespace::~\l_stex_module_ns_str\\
1379   Name::~\l_stex_module_name_str\\
1380   Language::~\l_stex_module_lang_str\\
1381   Signature::~\l_stex_module_sig_str\\
1382   Metatheory::~\l_stex_module_meta_str\\
1383   File::~\stex_path_to_string:N \g_stex_currentfile_seq
1384 }
1385
1386 \seq_put_right:Nx \l_stex_all_modules_seq {
1387   \l_stex_module_ns_str ? \l_stex_module_name_str
1388 }
1389
1390 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1391 %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1392
1393
1394 \stex_if_smsmode:F{
1395   \begin{stex_annotate_env} {theory} {
1396     \l_stex_module_ns_str ? \l_stex_module_name_str
1397   }
1398
1399   \stex_annotate_invisible:nnn{header}{} {
1400     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1401     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1402     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1403       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1404     }
1405     \str_if_empty:NF \smodulotype {
1406       \stex_annotate:nnn{type}{\smodulotype}{}
1407     }
1408   }
1409 }
1410 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1411 % TODO: Inherit metatheory for nested modules?
1412 }
1413 \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}

```

1414 \cs_new_protected:Nn \_stex_modules_end_module: {
1415   % \str_set:Nx \l_tmpa_str {
1416   %   c_stex_module_
1417   %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1418   %   \prop_item:Nn \l_stex_current_module_prop { name }
1419   %   _prop
1420   % }
1421   %%^A \prop_new:c { \l_tmpa_str }
1422   % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1423   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1424 }

```

(End definition for _stex_modules_end_module:.)

smodule The core environment, with no header

```

1425 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1426 \NewDocumentEnvironment { smodule } { 0{} m } {
1427   \stex_module_setup:nn{#1}{#2}
1428   \par
1429   \stex_if_smsmode:F{
1430     \tl_clear:N \l_tmpa_tl
1431     \clist_map_inline:Nn \smoduletype {
1432       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1433         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1434       }
1435     }
1436     \tl_if_empty:NTF \l_tmpa_tl {
1437       \__stex_modules_smodule_start:
1438     }{
1439       \l_tmpa_tl
1440     }
1441   }
1442   \__stex_modules_begin_module:
1443   \stex_ref_new_doc_target:n \smoduleid
1444   \stex_smsmode_do:
1445 } {
1446   \__stex_modules_end_module:
1447   \stex_if_smsmode:TF {
1448     % \exp_args:Nx \stex_add_to_sms:n {
1449     %   \prop_gset_from_keyval:cn {
1450     %     c_stex_module_
1451     %     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1452     %     \prop_item:Nn \l_stex_current_module_prop { name }
1453     %     _prop
1454     %   } {
1455     %     name      = \prop_item:cn { \l_tmpa_str } { name } ,
1456     %     ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1457     %     file      = \prop_item:cn { \l_tmpa_str } { file } ,
1458     %     lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1459     %     sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1460     %     meta      = \prop_item:cn { \l_tmpa_str } { meta }
1461     %   }
1462     % }
1463   }{
1464     \end{stex_annotate_env}
1465     \clist_set:No \l_tmpa_clist \smoduletype
1466     \tl_clear:N \l_tmpa_tl
1467     \clist_map_inline:Nn \l_tmpa_clist {
1468       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1469         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1470       }
1471     }
1472     \tl_if_empty:NTF \l_tmpa_tl {
1473       \__stex_modules_smodule_end:
1474     }{
1475       \l_tmpa_tl
1476     }
1477   }

```

```

1478 }
1479
1480 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1481 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1482
1483 \newcommand\stexpatchmodule[3] [] {
1484   \str_set:Nx \l_tmpa_str{ #1 }
1485   \str_if_empty:NTF \l_tmpa_str {
1486     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1487     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1488   }{
1489     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1490     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1491   }
1492 }
1493

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1494 \NewDocumentCommand \STEXModule { m } {
1495   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1496   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1497   \tl_set:Nn \l_tmpa_tl {
1498     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1499   }
1500   \seq_map_inline:Nn \l_stex_all_modules_seq {
1501     \str_set:Nn \l_tmpb_str { ##1 }
1502     \str_if_eq:eeT { \l_tmpa_str } {
1503       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1504     } {
1505       \seq_map_break:n {
1506         \tl_set:Nn \l_tmpa_tl {
1507           \stex_invoke_module:n { ##1 }
1508         }
1509       }
1510     }
1511   }
1512   \l_tmpa_tl
1513 }
1514
1515 \cs_new_protected:Nn \stex_invoke_module:n {
1516   \stex_debug:nn{modules}{Invoking~module~#1}
1517   \peek_charcode_remove:NTF ! {
1518     \__stex_modules_invoke_uri:nN { #1 }
1519   } {
1520     \peek_charcode_remove:NTF ? {
1521       \__stex_modules_invoke_symbol:nn { #1 }
1522     } {
1523       \msg_error:nnx{stex}{error/syntax}{
1524         ?~or~!~expected~after~
1525         \c_backslash_str STEXModule{#1}
1526       }
1527     }
1528   }
1529 }

```

```

1527     }
1528   }
1529 }
1530
1531 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1532   \str_set:Nn #2 { #1 }
1533 }
1534
1535 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1536   \stex_invoke_symbol:n{#1?#2}
1537 }

```

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

`\stex_activate_module:n`

```

1538 \bool_new:N \l_stex_in_meta_bool
1539 \bool_set_false:N \l_stex_in_meta_bool
1540 \cs_new_protected:Nn \stex_activate_module:n {
1541   \stex_debug:nn{modules}{Activating~module~#1}
1542   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1543     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1544   }
1545   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1546     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1547     \use:c{ c_stex_module_#1_code }
1548   }
1549 }

```

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

```

1550 </package>

```

Chapter 29

STEX -Module Inheritance Implementation

```
1551 <*package>
1552
1553 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1554
```

29.1 SMS Mode

```
1555 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1556 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1557 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1558 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1559
1560 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1561   \makeatletter
1562   \makeatother
1563   \ExplSyntaxOn
1564   \ExplSyntaxOff
1565   \rustexBREAK
1566 }
1567
1568 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1569   \symdef
1570   \importmodule
1571   \notation
1572   \symdecl
1573   \STEXexport
1574   \inlineass
1575   \inlinedef
1576   \inlineex
1577   \endinput
1578   \setnotation
```

```

1579 \copynotation
1580 }
1581
1582 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1583   \tl_to_str:n {
1584     smodule,
1585     copymodule,
1586     interpretmodule
1587     sdefinition,
1588     sexample,
1589     sassertion,
1590     sparagraph
1591   }
1592 }

```

(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`

```

1593 \bool_new:N \g__stex_smsmode_bool
1594 \bool_set_false:N \g__stex_smsmode_bool
1595 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1596   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1597 }

```

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

`\stex_in_smsmode:nn`

```

1598 \cs_new_protected:Nn \stex_in_smsmode:nn {
1599   \vbox_set:Nn \l_tmpa_box {
1600     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1601     \bool_gset_true:N \g__stex_smsmode_bool
1602     #2
1603     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1604   }
1605   \box_clear:N \l_tmpa_box
1606 }
1607
1608 \quark_new:N \q__stex_smsmode_break
1609
1610 %\ior_new:N \c__stex_smsmode_ior
1611 %\tl_new:N \l__stex_smsmode_filecontent_tl
1612 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1613   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1614   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1615   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1616     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1617   % }
1618   % \ior_close:N \c__stex_smsmode_ior
1619   \stex_filestack_push:n{#1}
1620   \stex_in_smsmode:nn{#1} {
1621     #2
1622     \everyeof{\q__stex_smsmode_break\noexpand}
1623     \expandafter\expandafter\expandafter
1624     \stex_smsmode_do:

```

```

1625     \csname @ @ input\endcsname "#1"\relax
1626     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1627   }
1628   \stex_filestack_pop:
1629 }

```

(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:

```

1630 \cs_new_protected:Npn \stex_smsmode_do: {
1631   \stex_if_smsmode:T {
1632     \__stex_smsmode_do:w
1633   }
1634 }
1635 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1636   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1637     \expandafter\if\expandafter\relax\noexpand#1
1638     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1639   } \else\expandafter\__stex_smsmode_do:w\fi
1640 }{
1641   \__stex_smsmode_do:w % #1
1642 }
1643 }
1644 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1645   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1646     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1647       #1\__stex_smsmode_do:w
1648     }{
1649       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1650         #1
1651       }{
1652         \cs_if_eq:NNTF \begin #1 {
1653           \__stex_smsmode_check_begin:n
1654         }{
1655           \cs_if_eq:NNTF \end #1 {
1656             \__stex_smsmode_check_end:n
1657           }{
1658             \__stex_smsmode_do:w
1659           }
1660         }
1661       }
1662     }
1663   }
1664 }
1665
1666 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1667   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1668     \begin{#1}
1669   }{
1670     \__stex_smsmode_do:w
1671   }
1672 }
1673 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1674 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1675   \end{#1}\__stex_smsmode_do:w
1676 }{
1677   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1678 }
1679 }

```

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

29.2 Inheritance

```

1680 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1681 \cs_new_protected:Nn \stex_import_module_uri:nn {
1682   \str_set:Nx \l_stex_import_archive_str { #1 }
1683   \str_set:Nn \l_stex_import_path_str { #2 }
1684
1685   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1686   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1687   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1688
1689   \stex_modules_current_namespace:
1690   \bool_lazy_all:nTF {
1691     {\str_if_empty_p:N \l_stex_import_archive_str}
1692     {\str_if_empty_p:N \l_stex_import_path_str}
1693     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1694   }{
1695     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1696     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1697   }{
1698     \str_if_empty:NT \l_stex_import_archive_str {
1699       \prop_if_exist:NT \l_stex_current_repository_prop {
1700         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1701       }
1702     }
1703     \str_if_empty:NTF \l_stex_import_archive_str {
1704       \str_if_empty:NF \l_stex_import_path_str {
1705         \str_set:Nx \l_stex_import_ns_str {
1706           \l_stex_module_ns_str / \l_stex_import_path_str
1707         }
1708       }
1709     }{
1710       \stex_require_repository:n \l_stex_import_archive_str
1711       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1712       \l_stex_import_ns_str
1713       \str_if_empty:NF \l_stex_import_path_str {
1714         \str_set:Nx \l_stex_import_ns_str {
1715           \l_stex_import_ns_str / \l_stex_import_path_str
1716         }
1717       }
1718     }
1719   }
1720 }

```

(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 1721 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1722 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1723 \str_new:N \l_stex_import_path_str
1724 \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>}
1725 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1726   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1727
1728     % archive
1729     \str_set:Nx \l_tmpa_str { #2 }
1730     \str_if_empty:NTF \l_tmpa_str {
1731       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1732     } {
1733       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1734       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1735       \seq_put_right:Nn \l_tmpa_seq { source }
1736     }
1737
1738     % path
1739     \str_set:Nx \l_tmpb_str { #3 }
1740     \str_if_empty:NTF \l_tmpb_str {
1741       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1742
1743       \ltx@ifpackageloaded{babel} {
1744         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1745           { \languagename } \l_tmpb_str {
1746           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1747         }
1748       } {
1749         \str_clear:N \l_tmpb_str
1750       }
1751
1752       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1753       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1754         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1755       }{
1756         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1757         \IfFileExists{ \l_tmpa_str.tex }{
1758           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1759         }{
1760           % try english as default
1761           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1762           \IfFileExists{ \l_tmpa_str.en.tex }{
1763             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1764           }{
1765             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1766           }
1767         }

```



```

1768     }
1769
1770   } {
1771     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1772     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1773
1774     \ltx@ifpackageloaded{babel} {
1775       \exp_args:Nnx \prop_get:NnNF \c_stex_language_abbrevs_prop
1776       { \language } \l_tmpb_str {
1777         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1778       }
1779     } {
1780       \str_clear:N \l_tmpb_str
1781     }
1782
1783     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1784
1785     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1786     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1787       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1788     }{
1789       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1790       \IfFileExists{ \l_tmpa_str/#4.tex }{
1791         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1792       }{
1793         % try english as default
1794         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1795         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1796           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1797         }{
1798           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1799           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1800             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1801           }{
1802             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1803             \IfFileExists{ \l_tmpa_str.tex }{
1804               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1805             }{
1806               % try english as default
1807               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1808               \IfFileExists{ \l_tmpa_str.en.tex }{
1809                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1810               }{
1811                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1812               }
1813             }
1814           }
1815         }
1816       }
1817     }
1818   }
1819
1820   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1821     \seq_clear:N \l_stex_all_modules_seq

```

```

1822     \str_clear:N \l_stex_current_module_str
1823     \str_set:Nx \l_tmpb_str { #2 }
1824     \str_if_empty:NF \l_tmpb_str {
1825         \stex_set_current_repository:n { #2 }
1826     }
1827     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1828 }
1829
1830 \stex_if_module_exists:nF { #1 ? #4 } {
1831     \msg_error:nnx{stex}{error/unknownmodule}{
1832         #1?#4~(in~file~\g__stex_importmodule_file_str)
1833     }
1834 }
1835 }
1836 \stex_activate_module:n { #1 ? #4 }
1837 }

```

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

`\importmodule`

```

1838 \NewDocumentCommand \importmodule { 0{} m } {
1839     \stex_import_module_uri:nn { #1 } { #2 }
1840     \stex_debug:nn{modules}{Importing~module:~
1841         \l_stex_import_ns_str ? \l_stex_import_name_str
1842     }
1843     \stex_if_smsmode:F {
1844         \stex_import_require_module:nnnn
1845         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1846         { \l_stex_import_path_str } { \l_stex_import_name_str }
1847         \stex_annotate_invisible:nnn
1848         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1849     }
1850     \exp_args:Nx \stex_add_to_current_module:n {
1851         \stex_import_require_module:nnnn
1852         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1853         { \l_stex_import_path_str } { \l_stex_import_name_str }
1854     }
1855     \exp_args:Nx \stex_add_import_to_current_module:n {
1856         \l_stex_import_ns_str ? \l_stex_import_name_str
1857     }
1858     \stex_smsmode_do:
1859     \ignorespacesandpars
1860 }
1861 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1862 \NewDocumentCommand \usemodule { 0{} m } {
1863     \stex_if_smsmode:F {
1864         \stex_import_module_uri:nn { #1 } { #2 }
1865         \stex_import_require_module:nnnn
1866         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1867         { \l_stex_import_path_str } { \l_stex_import_name_str }
1868         \stex_annotate_invisible:nnn

```

```

1869     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1870   }
1871   \stex_smsmode_do:
1872   \ignorespacesandpars
1873 }

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

1874 \</package>

```

Chapter 30

STEX -Symbols Implementation

```
1875 <*package>
1876
1877 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1878
Warnings and error messages
1879
```

30.1 Symbol Declarations

```
1880 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1881 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)

\STEXsymbol
1882 \NewDocumentCommand \STEXsymbol { m } {
1883   \stex_get_symbol:n { #1 }
1884   \exp_args:No
1885   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1886 }
(End definition for \STEXsymbol. This function is documented on page 38.)
symdecl arguments:
1887 \keys_define:nn { stex / symdecl } {
1888   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1889   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1890   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1891   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1892   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1893   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1894   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1895   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1896   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
```

```

1897 }
1898
1899 \bool_new:N \l_stex_symdecl_make_macro_bool
1900
1901 \cs_new_protected:Nn \__stex_symdecl_args:n {
1902   \str_clear:N \l_stex_symdecl_name_str
1903   \str_clear:N \l_stex_symdecl_args_str
1904   \str_clear:N \l_stex_symdecl_deprecate_str
1905   \bool_set_false:N \l_stex_symdecl_local_bool
1906   \tl_clear:N \l_stex_symdecl_type_tl
1907   \tl_clear:N \l_stex_symdecl_definiens_tl
1908
1909   \keys_set:nn { stex / symdecl } { #1 }
1910 }

```

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

```

1911
1912 \NewDocumentCommand \symdecl { s O{} m } {
1913   \__stex_symdecl_args:n { #2 }
1914   \IfBooleanTF #1 {
1915     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1916   } {
1917     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1918   }
1919   \stex_symdecl_do:n { #3 }
1920   \stex_smsmode_do:
1921 }
1922 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

\stex_symdecl_do:n

```

1923 \cs_new_protected:Nn \stex_symdecl_do:n {
1924   \stex_if_in_module:F {
1925     % TODO throw error? some default namespace?
1926   }
1927
1928   \str_if_empty:NT \l_stex_symdecl_name_str {
1929     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1930   }
1931
1932   \prop_if_exist:cT { l_stex_symdecl_
1933     \l_stex_current_module_str ?
1934     \l_stex_symdecl_name_str
1935   }_prop
1936 }{
1937   % TODO throw error (beware of circular dependencies)
1938 }
1939
1940 \prop_clear:N \l_tmpa_prop
1941 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1942 \seq_clear:N \l_tmpa_seq
1943 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str

```

```

1944 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1945
1946 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1947   \str_if_empty:NF \l_stex_module_deprecate_str {
1948     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1949   }
1950 }
1951 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1952
1953 \exp_args:No \stex_add_constant_to_current_module:n {
1954   \l_stex_symdecl_name_str
1955 }
1956
1957 % arity/args
1958 \int_zero:N \l_tmpb_int
1959
1960 \bool_set_true:N \l_tmpa_bool
1961 \str_map_inline:Nn \l_stex_symdecl_args_str {
1962   \token_case_meaning:NnF ##1 {
1963     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1964     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1965     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1966     {\tl_to_str:n a} {
1967       \bool_set_false:N \l_tmpa_bool
1968       \int_incr:N \l_tmpb_int
1969     }
1970     {\tl_to_str:n B} {
1971       \bool_set_false:N \l_tmpa_bool
1972       \int_incr:N \l_tmpb_int
1973     }
1974   }{
1975     \msg_set:nnn{stex}{error/wrongargs}{
1976       args~value~in~symbol~declaration~for~
1977       \l_stex_current_module_str ?
1978       \l_stex_symdecl_name_str ~
1979       needs~to~be~
1980       i,~a,~b~or~B,~but~##1~given
1981     }
1982     \msg_error:nn{stex}{error/wrongargs}
1983   }
1984 }
1985 \bool_if:NTF \l_tmpa_bool {
1986   % possibly numeric
1987   \str_if_empty:NTF \l_stex_symdecl_args_str {
1988     \prop_put:Nnn \l_tmpa_prop { args } {}
1989     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1990   }{
1991     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1992     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1993     \str_clear:N \l_tmpa_str
1994     \int_step_inline:nn \l_tmpa_int {
1995       \str_put_right:Nn \l_tmpa_str i
1996     }
1997     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }

```

```

1998     }
1999   } {
2000     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2001     \prop_put:Nnx \l_tmpa_prop { arity }
2002     { \str_count:N \l_stex_symdecl_args_str }
2003   }
2004   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2005
2006
2007   % semantic macro
2008
2009   \bool_if:NT \l_stex_symdecl_make_macro_bool {
2010     \exp_args:Nx \stex_do_aftergroup:n {
2011       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2012         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013       }}
2014     }
2015
2016     \bool_if:NF \l_stex_symdecl_local_bool {
2017       \exp_args:Nx \stex_add_to_current_module:n {
2018         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2019           \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020         } }
2021       }
2022     }
2023   }
2024
2025   % add to all symbols
2026
2027   \bool_if:NF \l_stex_symdecl_local_bool {
2028     \exp_args:Nx \stex_add_to_current_module:n {
2029       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2030         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2031       }
2032     }
2033   }
2034   % \exp_args:Nx \stex_add_field_to_current_module:n {
2035   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2036   % }
2037
2038   \stex_debug:nn{symbols}{New~symbol:~
2039     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2040     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2041     Args:~\prop_item:Nn \l_tmpa_prop { args }
2042   }
2043
2044   % circular dependencies require this:
2045
2046   \prop_if_exist:cF {
2047     l_stex_symdecl_
2048     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2049     _prop
2050   } {
2051     \prop_set_eq:cN {

```

```

2052     l_stex_symdecl_
2053     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2054     _prop
2055   } \l_tmpa_prop
2056 }
2057
2058 \seq_clear:c {
2059   l_stex_symdecl_
2060   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2061   _notations
2062 }
2063
2064 \bool_if:NF \l_stex_symdecl_local_bool {
2065   \exp_args:Nx
2066   \stex_add_to_current_module:n {
2067     \seq_clear:c {
2068       l_stex_symdecl_
2069       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2070       _notations
2071     }
2072     \prop_set_from_keyval:cn {
2073       l_stex_symdecl_
2074       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2075       _prop
2076     } {
2077       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2078       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2079       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2080       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2081       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2082       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2083     }
2084   }
2085 }
2086
2087 \stex_if_smsmode:TF {
2088   \bool_if:NF \l_stex_symdecl_local_bool {
2089     % \exp_args:Nx \stex_add_to_sms:n {
2090     %   \prop_set_from_keyval:cn {
2091     %     l_stex_symdecl_
2092     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2093     %     _prop
2094     %   } {
2095     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2096     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2097     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2098     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2099     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2100     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2101     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2102     %   }
2103     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2104     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2105     %   }

```



```

2106 %      }
2107     }
2108   }{
2109     \exp_args:Nx \stex_do_aftergroup:n {
2110       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2111         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2112       }
2113     }
2114     \stex_if_do_html:T {
2115       \stex_annotate_invisible:nnn {symdecl} {
2116         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2117       } {
2118         \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}{${\l_st
2119         \stex_annotate_invisible:nnn{args}{}{
2120           \prop_item:Nn \l_tmpa_prop { args }
2121         }
2122         \stex_annotate_invisible:nnn{macroname}{#1}{}
2123         \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2124           \stex_annotate_invisible:nnn{definiens}{}
2125             {${\l_stex_symdecl_definiens_tl$}
2126         }
2127       }
2128     }
2129   }
2130 }

```

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

`\stex_get_symbol:n`

```

2131 \str_new:N \l_stex_get_symbol_uri_str
2132
2133 \cs_new_protected:Nn \stex_get_symbol:n {
2134   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2135     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2136   }{
2137     % argument is a string
2138     % is it a command name?
2139     \cs_if_exist:cTF { #1 }{
2140       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2141       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2142       \str_if_empty:NTF \l_tmpa_str {
2143         \exp_args:Nx \cs_if_eq:NNTF {
2144           \tl_head:N \l_tmpa_tl
2145         } \stex_invoke_symbol:n {
2146           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2147         }{
2148           \__stex_symdecl_get_symbol_from_string:n { #1 }
2149         }
2150       } {
2151         \__stex_symdecl_get_symbol_from_string:n { #1 }
2152       }
2153     }{
2154       % argument is not a command name
2155       \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2156         % \l_stex_all_symbols_seq
2157     }
2158 }
2159 \str_if_eq:eeF {
2160     \prop_item:cn {
2161         l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2162     }{ deprecate }
2163 }{}{
2164     \msg_warning:nnxx{stex}{warning/deprecated}{
2165         Symbol~\l_stex_get_symbol_uri_str
2166     }{
2167         \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2168     }
2169 }
2170 }
2171
2172 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2173     \str_set:Nn \l_tmpa_str { #1 }
2174     \bool_set_false:N \l_tmpa_bool
2175     \stex_if_in_module:T {
2176         \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2177             \bool_set_true:N \l_tmpa_bool
2178             \str_set:Nx \l_stex_get_symbol_uri_str {
2179                 \l_stex_current_module_str ? #1
2180             }
2181         }
2182     }
2183     \bool_if:NF \l_tmpa_bool {
2184         \tl_set:Nn \l_tmpa_tl {
2185             \msg_set:nnn{stex}{error/unknownsymbol}{
2186                 No~symbol~#1~found!
2187             }
2188             \msg_error:nn{stex}{error/unknownsymbol}
2189         }
2190         \str_set:Nn \l_tmpa_str { #1 }
2191         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2192         \seq_map_inline:Nn \l_stex_all_symbols_seq {
2193             \str_set:Nn \l_tmpb_str { ##1 }
2194             \str_if_eq:eeT { \l_tmpa_str } {
2195                 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2196             } {
2197                 \seq_map_break:n {
2198                     \tl_set:Nn \l_tmpa_tl {
2199                         \str_set:Nn \l_stex_get_symbol_uri_str {
2200                             ##1
2201                         }
2202                     }
2203                 }
2204             }
2205         }
2206         \l_tmpa_tl
2207     }
2208 }
2209

```

```

2210 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2211   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2212     { \tl_tail:N \l_tmpa_tl }
2213   \tl_if_single:NTF \l_tmpa_tl {
2214     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2215       \exp_after:wN \str_set:Nn \exp_after:wN
2216         \l_stex_get_symbol_uri_str \l_tmpa_tl
2217     }{
2218       % TODO
2219       % tail is not a single group
2220     }
2221   }{
2222     % TODO
2223     % tail is not a single group
2224   }
2225 }

```

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

30.2 Notations

```

2226 <@@=stex_notation>
      notation arguments:
2227 \keys_define:nn { stex / notation } {
2228   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2229   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2230   prec .str_set_x:N = \l__stex_notation_prec_str ,
2231   op .tl_set:N = \l__stex_notation_op_tl ,
2232   primary .bool_set:N = \l__stex_notation_primary_bool ,
2233   primary .default:n = {true} ,
2234   unknown .code:n = \str_set:Nx
2235     \l__stex_notation_variant_str \l_keys_key_str
2236 }
2237
2238 \cs_new_protected:Nn \_stex_notation_args:n {
2239   \str_clear:N \l__stex_notation_lang_str
2240   \str_clear:N \l__stex_notation_variant_str
2241   \str_clear:N \l__stex_notation_prec_str
2242   \tl_clear:N \l__stex_notation_op_tl
2243   \bool_set_false:N \l__stex_notation_primary_bool
2244
2245   \keys_set:nn { stex / notation } { #1 }
2246 }

```

\notation

```

2247 \NewDocumentCommand \notation { 0{ } m } {
2248   \_stex_notation_args:n { #1 }
2249   \tl_clear:N \l_stex_symdecl_definiens_tl
2250   \stex_get_symbol:n { #2 }
2251   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2252 }
2253 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

`\stex_notation_do:nn`

```

2254 \seq_new:N \l__stex_notation_precedences_seq
2255 \tl_new:N \l__stex_notation_opprec_tl
2256 \int_new:N \l__stex_notation_currarg_int
2257
2258 \cs_new_protected:Nn \stex_notation_do:nn {
2259   \let\l_stex_current_symbol_str\relax
2260   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2261   \seq_clear:N \l__stex_notation_precedences_seq
2262   \tl_clear:N \l__stex_notation_opprec_tl
2263   \prop_get:cnN {
2264     l_stex_symdecl_ #1 _prop
2265   } { args } \l__stex_notation_args_str
2266
2267   % precedences
2268   \prop_get:cnN {
2269     l_stex_symdecl_ #1 _prop
2270   } { arity } \l__stex_notation_arity_str
2271   \str_if_empty:NTF \l__stex_notation_prec_str {
2272     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2273       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2274     }{
2275       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2276     }
2277   } {
2278     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2279       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2280       \int_step_inline:nn { \l__stex_notation_arity_str } {
2281         \exp_args:NNo
2282         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2283       }
2284     }{
2285       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2286       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2287         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2288         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2289           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2290             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2291           \seq_map_inline:Nn \l_tmpa_seq {
2292             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2293           }
2294         }
2295       }{
2296         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2297           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2298         }{
2299           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2300         }
2301       }
2302     }
2303   }
2304
2305   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2306   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2307 \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2308 \exp_args:NNo
2309 \seq_put_right:No \l__stex_notation_precedences_seq {
2310 \l__stex_notation_opprec_tl
2311 }
2312 }
2313 }
2314
2315 \tl_clear:N \l__stex_notation_dummyargs_tl
2316
2317 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2318 \exp_args:NNe
2319 \cs_set:Npn \l__stex_notation_macrocode_cs {
2320 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2321 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2322 { \l__stex_notation_opprec_tl }
2323 { \exp_not:n { #2 } }
2324 }
2325 \__stex_notation_final:
2326 }{
2327 \str_if_in:NnTF \l__stex_notation_args_str b {
2328 \exp_args:Nne \use:nn
2329 {
2330 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2331 \cs_set:Npn \l__stex_notation_arity_str } { {
2332 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2333 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2334 { \l__stex_notation_opprec_tl }
2335 { \exp_not:n { #2 } }
2336 } }
2337 }{
2338 \str_if_in:NnTF \l__stex_notation_args_str B {
2339 \exp_args:Nne \use:nn
2340 {
2341 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2342 \cs_set:Npn \l__stex_notation_arity_str } { {
2343 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2344 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2345 { \l__stex_notation_opprec_tl }
2346 { \exp_not:n { #2 } }
2347 } }
2348 }{
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_oma: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 }
2360

```

```

2361 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2362 \int_zero:N \l__stex_notation_currarg_int
2363 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2364 \__stex_notation_arguments:
2365 }
2366 }

```

(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

```

2367 \cs_new_protected:Nn \__stex_notation_arguments: {
2368 \int_incr:N \l__stex_notation_currarg_int
2369 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2370 \__stex_notation_final:
2371 }{
2372 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2373 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2374 \str_if_eq:VnTF \l_tmpa_str a {
2375 \__stex_notation_argument_assoc:n
2376 }{
2377 \str_if_eq:VnTF \l_tmpa_str B {
2378 \__stex_notation_argument_assoc:n
2379 }{
2380 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2381 \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2382 { \stex_term_math_arg:nnn
2383 { \int_use:N \l__stex_notation_currarg_int }
2384 { \l_tmpa_str }
2385 { ###\int_use:N \l__stex_notation_currarg_int }
2386 }
2387 }
2388 \__stex_notation_arguments:
2389 }
2390 }
2391 }
2392 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2393 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2394
2395 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2396 {\l__stex_notation_arity_str}{
2397 #1
2398 }
2399 \int_zero:N \l_tmpa_int
2400 \tl_clear:N \l_tmpa_tl
2401 \str_map_inline:Nn \l__stex_notation_args_str {
2402 \int_incr:N \l_tmpa_int
2403 \tl_put_right:Nx \l_tmpa_tl {
2404 \str_if_eq:nnTF {##1}{a}{ {} }{
2405 \str_if_eq:nnTF {##1}{B}{ {} }{
2406 {##### \int_use:N \l_tmpa_int}

```

```

2407     }
2408   }
2409 }
2410 }
2411 \exp_after:wN\exp_after:wN\exp_after:wN \def
2412 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2413 \exp_after:wN\exp_after:wN\exp_after:wN ##
2414 \exp_after:wN\exp_after:wN\exp_after:wN 1
2415 \exp_after:wN\exp_after:wN\exp_after:wN ##
2416 \exp_after:wN\exp_after:wN\exp_after:wN 2
2417 \exp_after:wN\exp_after:wN\exp_after:wN {
2418   \exp_after:wN \exp_after:wN \exp_after:wN
2419   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2420     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2421   }
2422 }
2423
2424 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2425 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2426   \stex_term_math_assoc_arg:nnnn
2427   { \int_use:N \l__stex_notation_currarg_int }
2428   { \l_tmpa_str }
2429   { ####\int_use:N \l__stex_notation_currarg_int }
2430   { \l_tmpa_cs {####1} {####2} }
2431 } }
2432 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2433 %\tl_put_right:Nx \l_tmpa_tl {
2434 % { \stex_term_math_assoc_arg:nnnn
2435 %   { \int_use:N \l_tmpa_int }
2436 %   { \l_tmpb_str }
2437 %   \exp_args:No \exp_not:n
2438 %   {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2439 %   { ####\int_use:N \l_tmpa_int }
2440 % }
2441 %}
2442 \__stex_notation_arguments:
2443 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2444 \cs_new_protected:Nn \__stex_notation_final: {
2445   \exp_args:Nne \use:nn
2446   {
2447     \cs_generate_from_arg_count:cNnn {
2448       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2449       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2450       _cs
2451     }
2452     \cs_set:Npn \l__stex_notation_arity_str } { {
2453       \exp_after:wN \exp_after:wN \exp_after:wN
2454       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2455       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2456     } }

```

```

2457
2458 \tl_if_empty:NF \l__stex_notation_op_tl {
2459   \cs_set:cpx {
2460     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2461     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2462     _cs
2463   } {
2464     \stex_term_oms:nnn {
2465       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2466       \l__stex_notation_lang_str
2467     }{
2468       \l__stex_notation_symbol_str
2469     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2470   }
2471 }
2472
2473 \exp_args:Ne
2474 \stex_add_to_current_module:n {
2475   \cs_generate_from_arg_count:cNnn {
2476     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2477     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2478     _cs
2479   } \cs_set:Npn {\l__stex_notation_arity_str} {
2480     \exp_after:wN \exp_after:wN \exp_after:wN
2481     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2482     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2483   }
2484   \tl_if_empty:NF \l__stex_notation_op_tl {
2485     \cs_set:cpn {
2486       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2487       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2488       _cs
2489     } {
2490       \stex_term_oms:nnn {
2491         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2492         \l__stex_notation_lang_str
2493       }{
2494         \l__stex_notation_symbol_str
2495       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2496     }
2497   }
2498 }
2499 \exp_args:Nx
2500 % \stex_do_aftergroup:n {
2501   \seq_put_right:cx {
2502     l_stex_symdecl_ \l__stex_notation_symbol_str
2503     _notations
2504   } {
2505     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2506   }
2507 % }
2508
2509 \stex_debug:nn{symbols}{
2510   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```



```

2511 ~for~\l__stex_notation_symbol_str^^J
2512 Operator~precedence::~\l__stex_notation_opprec_tl^^J
2513 Argument~precedences::~
2514   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2515 Notation: \cs_meaning:c {
2516   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2517   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2518   _cs
2519 }
2520 }
2521
2522 %\prop_set_eq:cn {
2523 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2524 %   \c_hash_str \l__stex_notation_lang_str _prop
2525 %} \l_tmpb_prop
2526
2527 \exp_args:Ne
2528 \stex_add_to_current_module:n {
2529   \seq_put_right:cn {
2530     l_stex_symdecl_ \l__stex_notation_symbol_str
2531     _notations
2532   } {
2533     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2534   }
2535   %\prop_set_from_keyval:cn {
2536   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2537   %   \c_hash_str \l__stex_notation_lang_str _prop
2538   %} {
2539   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2540   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2541   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2542   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2543   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2544   %}
2545 }
2546
2547 \stex_if_smsmode:TF {
2548 %   \exp_args:Nx \stex_add_to_sms:n {
2549 %     \prop_set_from_keyval:cn {
2550 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2551 %       \c_hash_str \l__stex_notation_lang_str _prop
2552 %     } {
2553 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2554 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2555 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2556 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2557 %       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2558 %     }
2559 %   }
2560 }{
2561
2562 % HTML annotations
2563 \stex_if_do_html:T {
2564   \stex_annotate_invisible:nnn { notation }

```

```

2565 { \l__stex_notation_symbol_str } {
2566   \stex_annotate_invisible:nnn { notationfragment }
2567   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2568   \stex_annotate_invisible:nnn { precedence }
2569   { \l__stex_notation_prec_str }{}
2570
2571   \int_zero:N \l_tmpa_int
2572   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2573   \tl_clear:N \l_tmpa_tl
2574   \int_step_inline:nn { \l__stex_notation_arity_str }{
2575     \int_incr:N \l_tmpa_int
2576     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2577     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2578     \str_if_eq:VnTF \l_tmpb_str a {
2579       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2580         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2581         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2582       } }
2583     }{
2584       \str_if_eq:VnTF \l_tmpb_str B {
2585         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2586           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2587           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2588         } }
2589       }{
2590         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2591           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2592         } }
2593       }
2594     }
2595   }
2596   \stex_annotate_invisible:nnn { notationcomp }{}{
2597     \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2598     $ \exp_args:Nno \use:nn { \use:c {
2599       stex_notation_ \l_stex_current_symbol_str
2600       \c_hash_str \l__stex_notation_variant_str
2601       \c_hash_str \l__stex_notation_lang_str _cs
2602     } } { \l_tmpa_tl } $
2603   }
2604 }
2605 }
2606 }
2607 \stex_smsmode_do:
2608 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

2609 \keys_define:nn { stex / setnotation } {
2610   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2611   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2612   unknown .code:n = \str_set:Nx
2613     \l__stex_notation_variant_str \l_keys_key_str
2614 }

```

```

2615
2616 \cs_new_protected:Nn \stex_setnotation_args:n {
2617   \str_clear:N \l__stex_notation_lang_str
2618   \str_clear:N \l__stex_notation_variant_str
2619   \keys_set:nn { stex / setnotation } { #1 }
2620 }
2621
2622 \NewDocumentCommand \setnotation {m m} {
2623   \stex_get_symbol:n { #1 }
2624   \stex_setnotation_args:n { #2 }
2625   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\l_stex_get_symbol_uri_str _notations }
2626     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2627     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notation
2628       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2629       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notation
2630         { \c_hash_str }
2631       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notations
2632         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2633       \exp_args:Nx \stex_add_to_current_module:n {
2634         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notati
2635           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2636         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notation
2637           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2638         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _notati
2639           { \c_hash_str }
2640       }
2641       \stex_debug:nn {notations}{
2642         Setting~default~notation~
2643         {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2644         \l_stex_get_symbol_uri_str \
2645         \expandafter\meaning\csname
2646         l_stex_symdecl\l_stex_get_symbol_uri_str _notations\endcsname
2647       }
2648     }{
2649       % todo throw error
2650     }
2651     \stex_smsmode_do:
2652 }
2653
2654 \cs_new_protected:Nn \stex_copy_notations:nn {
2655   \stex_debug:nn {notations}{
2656     Copying~notations~from~#2~to~#1\
2657     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2658   }
2659   \tl_clear:N \l_tmpa_tl
2660   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2661     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2662 }
2663 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2664   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2665   \edef \l_tmpa_tl {
2666     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2667     \exp_after:wN\exp_after:wN\exp_after:wN {
2668       \exp_after:wN \l_tmpa_cs \l_tmpa_tl

```

```

2669     }
2670   }
2671   \exp_args:Nx
2672   \stex_do_aftergroup:n {
2673     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2674     \cs_generate_from_arg_count:cNnn {
2675       stex_notation_ #1 \c_hash_str ##1 _cs
2676     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2677       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2678     }
2679   }
2680 }
2681 }
2682
2683 \NewDocumentCommand \copynotation {m m} {
2684   \stex_get_symbol:n { #1 }
2685   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2686   \stex_get_symbol:n { #2 }
2687   \exp_args:Noo
2688   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2689   \exp_args:Nx \stex_add_import_to_current_module:n{
2690     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2691   }
2692   \stex_smsmode_do:
2693 }
2694

```

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

\symdef

```

2695 \keys_define:nn { stex / symdef } {
2696   name .str_set_x:N = \l_stex_symdecl_name_str ,
2697   local .bool_set:N = \l_stex_symdecl_local_bool ,
2698   args .str_set_x:N = \l_stex_symdecl_args_str ,
2699   type .tl_set:N = \l_stex_symdecl_type_tl ,
2700   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2701   op .tl_set:N = \l__stex_notation_op_tl ,
2702   lang .str_set_x:N = \l__stex_notation_lang_str ,
2703   variant .str_set_x:N = \l__stex_notation_variant_str ,
2704   prec .str_set_x:N = \l__stex_notation_prec_str ,
2705   unknown .code:n = \str_set:Nx
2706     \l__stex_notation_variant_str \l_keys_key_str
2707 }
2708
2709 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2710   \str_clear:N \l_stex_symdecl_name_str
2711   \str_clear:N \l_stex_symdecl_args_str
2712   \bool_set_false:N \l_stex_symdecl_local_bool
2713   \tl_clear:N \l_stex_symdecl_type_tl
2714   \tl_clear:N \l_stex_symdecl_definiens_tl
2715   \str_clear:N \l__stex_notation_lang_str
2716   \str_clear:N \l__stex_notation_variant_str
2717   \str_clear:N \l__stex_notation_prec_str
2718   \tl_clear:N \l__stex_notation_op_tl

```

```

2719 \keys_set:nn { stex / symdef } { #1 }
2720 }
2721 }
2722
2723 \NewDocumentCommand \symdef { 0{} m } {
2724   \__stex_notation_symdef_args:n { #1 }
2725   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2726   \stex_symdecl_do:n { #2 }
2727   \exp_args:Nx \stex_notation_do:nn {
2728     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2729   }
2730 }
2731 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2732 \endpackage

```

Chapter 31

STEX -Terms Implementation

```
2733 <*package>
2734
2735 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2736
2737 <@@=stex_terms>
2738
2739 Warnings and error messages
2740 \msg_new:nnn{stex}{error/nonotation}{
2741   Symbol~#1~invoked,~but~has~no~notation~#2!
2742 }
2743 \msg_new:nnn{stex}{error/notationarg}{
2744   Error~in~parsing~notation~#1
2745 }
2746 \msg_new:nnn{stex}{error/noop}{
2747   Symbol~#1~has~no~operator~notation~for~notation~#2
2748 }
```

31.1 Symbol Invocations

Arguments:

```
2748 \keys_define:nn { stex / terms } {
2749   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2750   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2751   unknown .code:n = \str_set:Nx
2752     \l__stex_terms_variant_str \l_keys_key_str
2753 }
2754
2755 \cs_new_protected:Nn \__stex_terms_args:n {
2756   \str_clear:N \l__stex_terms_lang_str
2757   \str_clear:N \l__stex_terms_variant_str
2758   \str_clear:N \l__stex_terms_prec_str
2759   \tl_clear:N \l__stex_terms_op_tl
2760
2761   \keys_set:nn { stex / terms } { #1 }
```

2762 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2763 \cs_new_protected:Nn \stex_invoke_symbol:n {
2764   \str_if_eq:eeF {
2765     \prop_item:cn {
2766       l_stex_symdecl_#1_prop
2767     }{ deprecate }
2768   }{}{
2769     \msg_warning:nxxx{stex}{warning/deprecated}{
2770       Symbol~#1
2771     }{
2772       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2773     }
2774   }
2775   \if_mode_math:
2776     \exp_after:wN \__stex_terms_invoke_math:n
2777   \else:
2778     \exp_after:wN \__stex_terms_invoke_text:n
2779   \fi: { #1 }
2780 }
```

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

`__stex_terms_invoke_math:n`

```

2781 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2782   \peek_charcode_remove:NTF ! {
2783     \peek_charcode:NTF [ {
2784       \__stex_terms_invoke_op:nw { #1 }
2785     }{
2786       \peek_charcode_remove:NTF ! {
2787         \peek_charcode:NTF [ {
2788           \__stex_terms_invoke_op_custom:nw
2789         }{
2790           % TODO throw error
2791         }
2792       }{
2793         \__stex_terms_invoke_op:nw { #1 } []
2794       }
2795     }
2796   }{
2797     \peek_charcode_remove:NTF * {
2798       \__stex_terms_invoke_text:n { #1 }
2799     }{
2800       \peek_charcode:NTF [ {
2801         \__stex_terms_invoke_math:nw { #1 }
2802       }{
2803         \__stex_terms_invoke_math:nw { #1 } []
2804       }
2805     }
2806   }
2807 }
```

(End definition for `__stex_terms_invoke_math:n`.)

_stex_terms_invoke_op_custom:nw

```

2808 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2809   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2810     \stex_highlight_term:nn{#1}{#2}
2811   }
2812 }

```

(End definition for _stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2813 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2814   \_stex_terms_args:n { #2 }
2815   \cs_if_exist:cTF {
2816     stex_op_notation_ #1 \c_hash_str
2817     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2818   }{
2819     \csname stex_op_notation_ #1 \c_hash_str
2820       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2821     \endcsname
2822   }{
2823     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2824   }
2825 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2826 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2827   \_stex_terms_args:n { #2 }
2828   \seq_if_empty:cTF {
2829     l_stex_symdecl_ #1 _notations
2830   } {
2831     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2832   } {
2833     \seq_if_in:cxTF {
2834       l_stex_symdecl_ #1 _notations
2835     }
2836     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2837       \str_set:Nn \l_stex_current_symbol_str { #1 }
2838       \stex_debug:nn{terms}{Using~
2839         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2840       \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2841       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2842       _cs\endcsname
2843     }
2844     \use:c{
2845       stex_notation_ #1 \c_hash_str
2846       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2847       _cs
2848     }
2849   }{
2850     \str_if_empty:NTF \l__stex_terms_variant_str {
2851       \str_if_empty:NTF \l__stex_terms_lang_str {
2852         \seq_get_left:cN {

```



```

2853         l_stex_symdecl_ #1 _notations
2854     } \l_tmpa_str
2855     \str_set:Nn \l_stex_current_symbol_str { #1 }
2856     \stex_debug:nn{terms}{Using~
2857         #1\c_hash_str\l_tmpa_str \\
2858         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2859         \l_tmpa_str
2860         _cs\endcsname
2861     }
2862     \use:c{
2863         stex_notation_ #1 \c_hash_str \l_tmpa_str
2864         _cs
2865     }
2866   }{
2867     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2868       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2869     }
2870   }
2871 }{
2872   \msg_error:nnxx{stex}{error/nonotation}{#1}{
2873     ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2874   }
2875 }
2876 }
2877 }
2878 }

```

(End definition for _stex_terms_invoke_math:nw.)

_stex_terms_invoke_text:n

```

2879 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2880   \peek_charcode_remove:NTF ! {
2881     \stex_term_custom:nn { #1 } { }
2882   }{
2883     \prop_set_eq:Nc \l_tmpa_prop {
2884       l_stex_symdecl_ #1 _prop
2885     }
2886     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2887     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2888   }
2889 }

```

(End definition for _stex_terms_invoke_text:n.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2890 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2891 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2892 \int_new:N \l__stex_terms_downprec
2893 \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
```

```
2894 \tl_set:Nn \l__stex_terms_left_bracket_str (
2895 \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

```
2896 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2897   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2898     \bool_set_false:N \l__stex_terms_brackets_done_bool
2899     #2
2900   } {
2901     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2902       \bool_if:NTF \l_stex_inarray_bool { #2 } {
2903         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2904         \dobrackets { #2 }
2905       }
2906     }{ #2 }
2907   }
2908 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

\dobrackets

```
2909 \bool_new:N \l__stex_terms_brackets_done_bool
2910 %\RequirePackage{scalerel}
2911 \cs_new_protected:Npn \dobrackets #1 {
2912   %\ThisStyle{\if D\m@switch
2913   %   \exp_args:Nnx \use:nn
2914   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2915   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2916   %   \else
2917   %   \exp_args:Nnx \use:nn
2918   %   {
2919     \bool_set_true:N \l__stex_terms_brackets_done_bool
2920     \int_set:Nn \l__stex_terms_downprec \infprec
2921     \l__stex_terms_left_bracket_str
2922     #1
2923   }
2924   {
2925     \bool_set_false:N \l__stex_terms_brackets_done_bool
2926     \l__stex_terms_right_bracket_str
2927     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2928   }
2929   %\fi}
2930 }
```

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

\withbrackets

```
2931 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2932   \exp_args:Nnx \use:nn
2933   {
2934     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2935     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2936     #3
2937   }
2938   {
2939     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2940     {\l__stex_terms_left_bracket_str}
2941     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2942     {\l__stex_terms_right_bracket_str}
2943   }
2944 }
```

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

\STEXinvisible

```
2945 \cs_new_protected:Npn \STEXinvisible #1 {
2946   \stex_annotate_invisible:n { #1 }
2947 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
2948 \cs_new_protected:Nn \_stex_term_oms:nnn {
2949   \stex_annotate:nnn{ OMID }{ #2 }{
2950     \stex_highlight_term:nn { #1 } { #3 }
2951   }
2952 }
2953
2954 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2955   \__stex_terms_maybe_brackets:nn { #3 }{
2956     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2957   }
2958 }
```

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

_stex_term_math_oma:nnnn

```
2959 \cs_new_protected:Nn \_stex_term_oma:nnn {
2960   \stex_annotate:nnn{ OMA }{ #2 }{
2961     \stex_highlight_term:nn { #1 } { #3 }
2962   }
2963 }
2964
2965 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2966   \__stex_terms_maybe_brackets:nn { #3 }{
2967     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2968   }
2969 }
```

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

`_stex_term_math_omb:nnnn`

```
2970 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2971   \stex_annotate:nnn{ OMBIND }{ #2 }{
2972     \stex_highlight_term:nn { #1 } { #3 }
2973   }
2974 }
2975
2976 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2977   \__stex_terms_maybe_brackets:nn { #3 }{
2978     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2979   }
2980 }
```

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

`_stex_term_math_arg:nnn`

```
2981 \cs_new_protected:Nn \_stex_term_arg:nn {
2982   \stex_unhighlight_term:n {
2983     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2984   }
2985 }
2986 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2987   \exp_args:Nnx \use:nn
2988     { \int_set:Nn \l__stex_terms_downprec { #2 }
2989       \_stex_term_arg:nn { #1 }{ #3 }
2990     }
2991     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2992   }
2993 }
```

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

`_stex_term_math_assoc_arg:nnnn`

```
2993 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2994   % TODO sequences
2995   \clist_set:Nn \l_tmpa_clist{ #3 }
2996   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2997     \tl_set:Nn \l_tmpa_tl { #3 }
2998   }{
2999     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3000     \clist_reverse:N \l_tmpa_clist
3001     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3002
3003     \clist_map_inline:Nn \l_tmpa_clist {
3004       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3005         \exp_args:Nno
3006         \l_tmpa_cs { ##1 } \l_tmpa_tl
3007       }
3008     }
3009   }
3010   \exp_args:Nnno
3011     \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3012 }
```

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

`\stex_term_custom:nn`

```

3013 \cs_new_protected:Nn \stex_term_custom:nn {
3014   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3015   \str_set:Nn \l_tmpa_str { #2 }
3016   \tl_clear:N \l_tmpa_tl
3017   \int_zero:N \l_tmpa_int
3018   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3019   \__stex_terms_custom_loop:
3020 }

```

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

`__stex_terms_custom_loop:`

```

3021 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3022   \bool_set_false:N \l_tmpa_bool
3023   \bool_while_do:nn {
3024     \str_if_eq_p:ee X {
3025       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3026     }
3027   }{
3028     \int_incr:N \l_tmpa_int
3029   }
3030
3031   \peek_charcode:NTF [ {
3032     % notation/text component
3033     \__stex_terms_custom_component:w
3034   } {
3035     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3036       % all arguments read => finish
3037       \__stex_terms_custom_final:
3038     } {
3039       % arguments missing
3040       \peek_charcode_remove:NTF * {
3041         % invisible, specific argument position or both
3042         \peek_charcode:NTF [ {
3043           % visible specific argument position
3044           \__stex_terms_custom_arg:wn
3045         } {
3046           % invisible
3047           \peek_charcode_remove:NTF * {
3048             % invisible specific argument position
3049             \__stex_terms_custom_arg_inv:wn
3050           } {
3051             % invisible next argument
3052             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3053           }
3054         }
3055       } {
3056         % next normal argument
3057         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3058       }
3059     }
3060   }
3061 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```
3062 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3063   \bool_set_true:N \l_tmpa_bool
3064   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3065 }
```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```
3066 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3067   \str_set:Nx \l_tmpb_str {
3068     \str_item:Nn \l_tmpa_str { #1 }
3069   }
3070   \str_case:VnTF \l_tmpb_str {
3071     { X } {
3072       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3073     }
3074     { i } { \_stex_terms_custom_set_X:n { #1 } }
3075     { b } { \_stex_terms_custom_set_X:n { #1 } }
3076     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3077     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3078   }{}{
3079     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3080   }
3081
3082   \bool_if:nTF \l_tmpa_bool {
3083     \tl_put_right:Nx \l_tmpa_tl {
3084       \stex_annotate_invisible:n {
3085         \_stex_term_arg:nn { \int_eval:n { #1 } }
3086         \exp_not:n { { #2 } }
3087       }
3088     }
3089   } {
3090     \tl_put_right:Nx \l_tmpa_tl {
3091       \_stex_term_arg:nn { \int_eval:n { #1 } }
3092       \exp_not:n { { #2 } }
3093     }
3094   }
3095
3096   \_stex_terms_custom_loop:
3097 }
```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```
3098 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3099   \str_set:Nx \l_tmpa_str {
3100     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3101     X
3102     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3103   }
3104 }
```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

3105 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3106   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3107   \_stex_terms_custom_loop:
3108 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

3109 \cs_new_protected:Nn \_stex_terms_custom_final: {
3110   \int_compare:nNnTF \l_tmpb_int = 0 {
3111     \exp_args:Nnno \_stex_term_oms:nnn
3112   }{
3113     \str_if_in:NnTF \l_tmpa_str {b} {
3114       \exp_args:Nnno \_stex_term_ombind:nnn
3115     } {
3116       \exp_args:Nnno \_stex_term_oma:nnn
3117     }
3118   }
3119   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3120 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

3121 \NewDocumentCommand \symref { m m }{
3122   \let\compemph_uri_prev:\compemph@uri
3123   \let\compemph@uri\symrefemph@uri
3124   \STEXsymbol{#1}!{#2}
3125   \let\compemph@uri\compemph_uri_prev:
3126 }
3127
3128 \keys_define:nn { stex / symname } {
3129   post .str_set_x:N = \l_stex_symname_post_str
3130 }
3131
3132 \cs_new_protected:Nn \stex_symname_args:n {
3133   \str_clear:N \l_stex_symname_post_str
3134   \keys_set:nn { stex / symname } { #1 }
3135 }
3136
3137 \NewDocumentCommand \symname { 0{ } m }{
3138   \stex_symname_args:n { #1 }
3139   \stex_get_symbol:n { #2 }
3140   \str_set:Nx \l_tmpa_str {
3141     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3142   }
3143   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3144
3145   \let\compemph_uri_prev:\compemph@uri
3146   \let\compemph@uri\symrefemph@uri
3147   \exp_args:NNx \use:nn

```

```

3148 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3149 \l_tmpa_str \l_stex_symname_post_str
3150 ] }
3151 \let\compemph@uri\compemph_uri_prev:
3152 }

```

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

31.3 Notation Components

```

3153 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3154
3155 \str_new:N \l_stex_current_symbol_str
3156 \cs_new_protected:Nn \stex_highlight_term:nn {
3157   \exp_args:Nnx
3158   \use:nn {
3159     \str_set:Nx \l_stex_current_symbol_str { #1 }
3160     #2
3161   } {
3162     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3163     { \l_stex_current_symbol_str }
3164   }
3165 }
3166
3167 \cs_new_protected:Nn \stex_unhighlight_term:n {
3168   % \latexml_if:TF {
3169   %   #1
3170   % } {
3171   %   \rustex_if:TF {
3172   %     #1
3173   %   } {
3174     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3175   % }
3176   % }
3177 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3178 \cs_new_protected:Npn \comp #1 {
3179   \str_if_empty:NF \l_stex_current_symbol_str {
3180     \rustex_if:TF {
3181       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3182     }{
3183       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3184     }
3185   }
3186 }
3187
3188 \cs_new_protected:Npn \compemph@uri #1 #2 {
3189   \compemph{ #1 }
3190 }

```



```

3191
3192
3193 \cs_new_protected:Npn \compemph #1 {
3194     #1
3195 }
3196
3197 \cs_new_protected:Npn \defemph@uri #1 #2 {
3198     \defemph{#1}
3199 }
3200
3201 \cs_new_protected:Npn \defemph #1 {
3202     \textbf{#1}
3203 }
3204
3205 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3206     \symrefemph{#1}
3207 }
3208
3209 \cs_new_protected:Npn \symrefemph #1 {
3210     \textbf{#1}
3211 }

```

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

`\ellipses`

```

3212 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3213 \bool_new:N \l_stex_inarray_bool
\parrayline 3214 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3215 \NewDocumentCommand \parray { m m } {
\parraycell 3216     \begingroup
3217     \bool_set_true:N \l_stex_inarray_bool
3218     \begin{array}{#1}
3219         #2
3220     \end{array}
3221     \endgroup
3222 }
3223
3224 \NewDocumentCommand \prmatrix { m } {
3225     \begingroup
3226     \bool_set_true:N \l_stex_inarray_bool
3227     \begin{matrix}
3228         #1
3229     \end{matrix}
3230     \endgroup
3231 }
3232
3233 \def \maybepline {
3234     \bool_if:NT \l_stex_inarray_bool {\hline}
3235 }
3236
3237 \def \parrayline #1 #2 {

```

```

3238   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\}
3239 }
3240
3241 \def \pmrow #1 { \parrayline{ }{ #1 } }
3242
3243 \def \parraylineh #1 #2 {
3244   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\hline}
3245 }
3246
3247 \def \parraycell #1 {
3248   #1 \bool_if:NT \l_stex_inarray_bool {&}
3249 }

```

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

```

3250 \endpackage

```

Chapter 32

STEX -Structural Features Implementation

```
3251 <*package>
3252
3253 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3254
3255 <@@=stex_features>
3256
3257 Warnings and error messages
3258 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3259   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3260 }
3261 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3262   Symbol~#1~not~assigned~in~interpretmodule~#2
3263 }
```

32.1 Imports with modification

```
3263 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3264   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3265     \__stex_features_get_symbol_from_cs:n { #1 }
3266   }{
3267     % argument is a string
3268     % is it a command name?
3269     \cs_if_exist:cTF { #1 }{
3270       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3271       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3272       \str_if_empty:NNTF \l_tmpa_str {
3273         \exp_args:Nx \cs_if_eq:NNTF {
3274           \tl_head:N \l_tmpa_tl
3275         } \stex_invoke_symbol:n {
3276           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3277         }{
3278           \__stex_features_get_symbol_from_string:n { #1 }
3279         }
3280       }
3281     }
3282   }
```

```

3279     }
3280   } {
3281     \__stex_features_get_symbol_from_string:n { #1 }
3282   }
3283   ){
3284     % argument is not a command name
3285     \__stex_features_get_symbol_from_string:n { #1 }
3286     % \l_stex_all_symbols_seq
3287   }
3288 }
3289 }
3290
3291 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3292   \str_set:Nn \l_tmpa_str { #1 }
3293   \bool_set_false:N \l_tmpa_bool
3294   \bool_if:NF \l_tmpa_bool {
3295     \tl_set:Nn \l_tmpa_tl {
3296       \msg_set:nnn{stex}{error/unknownsymbol}{
3297         No~symbol~#1~found!
3298       }
3299       \msg_error:nn{stex}{error/unknownsymbol}
3300     }
3301     \str_set:Nn \l_tmpa_str { #1 }
3302     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3303     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3304       \str_set:Nn \l_tmpb_str { ##1 }
3305       \str_if_eq:eeT { \l_tmpa_str } {
3306         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3307       } {
3308         \seq_map_break:n {
3309           \tl_set:Nn \l_tmpa_tl {
3310             \str_set:Nn \l_stex_get_symbol_uri_str {
3311               ##1
3312             }
3313             \__stex_features_get_symbol_check:
3314           }
3315         }
3316       }
3317     }
3318     \l_tmpa_tl
3319   }
3320 }
3321
3322 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3323   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3324   { \tl_tail:N \l_tmpa_tl }
3325   \tl_if_single:NTF \l_tmpa_tl {
3326     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3327       \exp_after:wN \str_set:Nn \exp_after:wN
3328       \l_stex_get_symbol_uri_str \l_tmpa_tl
3329       \__stex_features_get_symbol_check:
3330     }{
3331       % TODO
3332       % tail is not a single group

```

```

3333     }
3334   }{
3335     % TODO
3336     % tail is not a single group
3337   }
3338 }
3339
3340 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3341   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3342   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3343     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3344     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3345     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3346       \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3347         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3348       }
3349     }
3350   }{
3351     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3352       \l_stex_current_copymodule_name_str~(inexplicably)
3353     }
3354   }
3355 }
3356
3357 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3358   \stex_import_module_uri:nn { #1 } { #2 }
3359   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3360   \stex_import_require_module:nnnn
3361     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3362     { \l_stex_import_path_str } { \l_stex_import_name_str }
3363   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3364   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3365   \seq_clear:N \l__stex_features_copymodule_fields_seq
3366   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3367     \seq_map_inline:cn {c_stex_module_###1_constants}{
3368       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3369         ###1 ? #####1
3370       }
3371     }
3372   }
3373   \seq_clear:N \l_tmpa_seq
3374   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3375     name      = \l_stex_current_copymodule_name_str ,
3376     module    = \l_stex_current_module_str ,
3377     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3378     includes  = \l_tmpa_seq ,
3379     fields    = \l_tmpa_seq
3380   }
3381   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3382     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3383   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3384     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3385   \stex_if_smsmode:F {
3386     \begin{stex_annotate_env} {#4} {

```

```

3387     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3388   }
3389   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3390 }
3391 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3392 \bool_set_false:N \l_stex_html_do_output_bool
3393 }
3394 \cs_new_protected:Nn \stex_copymodule_end:n {
3395   \def \l_tmpa_cs ##1 ##2 {#1}
3396   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3397   \tl_clear:N \l_tmpa_tl
3398   \tl_clear:N \l_tmpb_tl
3399   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3400   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3401     \seq_map_inline:cn {c_stex_module_##1_constants}{
3402       \tl_clear:N \l_tmpc_tl
3403       \l_tmpa_cs{##1}{####1}
3404       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3405         \tl_put_right:Nx \l_tmpa_tl {
3406           \prop_set_from_keyval:cn {
3407             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3408           }{
3409             \exp_after:wN \prop_to_keyval:N \csname
3410               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3411             \endcsname
3412           }
3413           \seq_clear:c {
3414             l_stex_symdecl_
3415             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3416             _notations
3417           }
3418         }
3419         \tl_put_right:Nx \l_tmpc_tl {
3420           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3421           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3422         }
3423         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3424         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3425           \tl_put_right:Nx \l_tmpc_tl {
3426             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3427           }
3428           \tl_put_right:Nx \l_tmpa_tl {
3429             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\}
3430             \stex_invoke_symbol:n {
3431               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3432             }
3433           }
3434         }
3435       }
3436     }{
3437       \tl_put_right:Nx \l_tmpc_tl {
3438         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3439       }
3440       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3441 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3442 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3443 \tl_put_right:Nx \l_tmpa_tl {
3444   \prop_set_from_keyval:cn {
3445     l_stex_symdecl \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3446   }{
3447     \prop_to_keyval:N \l_tmpa_prop
3448   }
3449   \seq_clear:c {
3450     l_stex_symdecl_
3451     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3452     _notations
3453   }
3454 }
3455 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3456 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3457   \tl_put_right:Nx \l_tmpc_tl {
3458     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3459   }
3460   \tl_put_right:Nx \l_tmpa_tl {
3461     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3462       \stex_invoke_symbol:n {
3463         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3464       }
3465     }
3466   }
3467 }
3468 }
3469 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3470   \tl_put_right:Nx \l_tmpc_tl {
3471     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3472   }
3473 }
3474 \tl_put_right:Nx \l_tmpb_tl {
3475   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3476 }
3477 }
3478 }
3479 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3480 \tl_put_left:Nx \l_tmpa_tl {
3481   \prop_set_from_keyval:cn {
3482     l_stex_copymodule_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str _prop
3483   }{
3484     \prop_to_keyval:N \l_stex_current_copymodule_prop
3485   }
3486 }
3487 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3488 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3489 \exp_args:Nx \stex_do_aftergroup:n {
3490   \exp_args:No \exp_not:n \l_tmpa_tl
3491 }
3492 \l_tmpb_tl
3493 \stex_if_smsmode:F {
3494   \end{stex_annotate_env}

```

```

3495 }
3496 }
3497
3498 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3499   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3500   \stex_deactivate_macro:Nn \symdecl {module~environments}
3501   \stex_deactivate_macro:Nn \symdef {module~environments}
3502   \stex_deactivate_macro:Nn \notation {module~environments}
3503   \stex_reactivate_macro:N \assign
3504   \stex_reactivate_macro:N \renamedec1
3505   \stex_reactivate_macro:N \donotcopy
3506   \stex_smsmode_do:
3507 }{
3508   \stex_copymodule_end:n {}
3509 }
3510
3511 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3512   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3513   \stex_deactivate_macro:Nn \symdecl {module~environments}
3514   \stex_deactivate_macro:Nn \symdef {module~environments}
3515   \stex_deactivate_macro:Nn \notation {module~environments}
3516   \stex_reactivate_macro:N \assign
3517   \stex_reactivate_macro:N \renamedec1
3518   \stex_reactivate_macro:N \donotcopy
3519   \stex_smsmode_do:
3520 }{
3521   \stex_copymodule_end:n {
3522     \tl_if_exist:cF {
3523       l__stex_features_copymodule_##1?##2_def_tl
3524     }{
3525       \msg_error:nnxx{stex}{error/interpretmodule/nodéfiniens}{
3526         ##1?##2
3527       }{\l_stex_current_copymodule_name_str}
3528     }
3529   }
3530 }
3531
3532 \NewDocumentCommand \donotcopy { 0{} m}{
3533   \stex_import_module_uri:nn { #1 } { #2 }
3534   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3535   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3536     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3537     \seq_map_inline:cn {c_stex_module_##1_constants}{
3538       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3539       \bool_lazy_any_p:nT {
3540         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3541         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3542         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3543       }{
3544         % TODO throw error
3545       }
3546     }
3547   }
3548 }

```



```

3549 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3550 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3551 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3552 }
3553
3554 \NewDocumentCommand \assign { m m }{
3555   \stex_get_symbol_in_copymodule:n {#1}
3556   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3557   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3558 }
3559
3560 \keys_define:nn { stex / renamedec1 } {
3561   name .str_set_x:N = \l_stex_renamedec1_name_str
3562 }
3563 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3564   \str_clear:N \l_stex_renamedec1_name_str
3565
3566   \keys_set:nn { stex / renamedec1 } { #1 }
3567 }
3568
3569 \NewDocumentCommand \renamedec1 { 0{} m m }{
3570   \__stex_features_renamedec1_args:n { #1 }
3571   \stex_get_symbol_in_copymodule:n {#2}
3572   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3573   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3574   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3575     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3576       \l_stex_get_symbol_uri_str
3577     } }
3578   } {
3579     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3580     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3581     \prop_set_eq:cc {l_stex_symdecl_
3582       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3583       _prop
3584     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3585     \seq_set_eq:cc {l_stex_symdecl_
3586       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3587       _notations
3588     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3589     \prop_put:cnx {l_stex_symdecl_
3590       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3591       _prop
3592     }{ name }{ \l_stex_renamedec1_name_str }
3593     \prop_put:cnx {l_stex_symdecl_
3594       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3595       _prop
3596     }{ module }{ \l_stex_current_module_str }
3597     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3598       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3599     }
3600     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3601       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3602     } }

```

```

3603 }
3604 }
3605 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3606 % \_stex_notation_args:n { #1 }
3607 % \tl_clear:N \l_stex_symdecl_definiens_tl
3608 % \stex_get_symbol_in_copymodule:n { #2 }
3609 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3610 % % todo
3611 %}
3612 \stex_deactivate_macro:Nn \assign {copymodules}
3613 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3614 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3615
3616
3617 \seq_new:N \l_stex_implicit_morphisms_seq
3618 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3619 \stex_import_module_uri:nn { #1 } { #2 }
3620 \stex_debug:nn{implicits}{
3621 Implicit~morphism:~
3622 \l_stex_module_ns_str ? \l__stex_features_name_str
3623 }
3624 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3625 \l_stex_module_ns_str ? \l__stex_features_name_str
3626 }{
3627 \msg_error:nnn{stex}{error/conflictingmodules}{
3628 \l_stex_module_ns_str ? \l__stex_features_name_str
3629 }
3630 }
3631
3632 % TODO
3633
3634
3635
3636 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3637 \l_stex_module_ns_str ? \l__stex_features_name_str
3638 }
3639 }
3640

```

32.2 The feature environment

structural@feature

```

3641
3642 \NewDocumentEnvironment{structural@feature}{ m m m }{
3643 \stex_if_in_module:F {
3644 \msg_set:nnn{stex}{error/nomodule}{
3645 Structural~Feature~has~to~occur~in~a~module:\\
3646 Feature~#2~of~type~#1\\
3647 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3648 }
3649 \msg_error:nn{stex}{error/nomodule}
3650 }
3651

```

```

3652 \str_set:Nx \l_stex_module_name_str {
3653   \prop_item:Nn \l_stex_current_module_prop
3654     { name } / #2 - feature
3655 }
3656
3657 \str_set:Nx \l_stex_module_ns_str {
3658   \prop_item:Nn \l_stex_current_module_prop
3659     { ns }
3660 }
3661
3662
3663 \str_clear:N \l_tmpa_str
3664 \seq_clear:N \l_tmpa_seq
3665 \tl_clear:N \l_tmpa_tl
3666 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3667   origname = #2,
3668   name     = \l_stex_module_name_str ,
3669   ns       = \l_stex_module_ns_str ,
3670   imports  = \exp_not:o { \l_tmpa_seq } ,
3671   constants = \exp_not:o { \l_tmpa_seq } ,
3672   content  = \exp_not:o { \l_tmpa_tl } ,
3673   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3674   lang     = \l_stex_module_lang_str ,
3675   sig      = \l_tmpa_str ,
3676   meta     = \l_tmpa_str ,
3677   feature  = #1 ,
3678 }
3679
3680 \stex_if_smsmode:F {
3681   \begin{stex_annotate_env}{ feature:#1 }{}
3682   \stex_annotate_invisible:nnn{header}{}{ #3 }
3683 }
3684 }{
3685   \str_set:Nx \l_tmpa_str {
3686     c_stex_feature_
3687     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3688     \prop_item:Nn \l_stex_current_module_prop { name }
3689     _prop
3690   }
3691   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3692   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3693   \stex_if_smsmode:TF {
3694     \exp_args:Nx \stex_add_to_sms:n {
3695       \prop_gset_from_keyval:cn {
3696         c_stex_feature_
3697         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3698         \prop_item:Nn \l_stex_current_module_prop { name }
3699         _prop
3700       } {
3701         origname = #2,
3702         name     = \prop_item:cn { \l_tmpa_str } { name } ,
3703         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
3704         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3705         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3706         content    = \prop_item:cn { \l_tmpa_str } { content } ,
3707         file        = \prop_item:cn { \l_tmpa_str } { file } ,
3708         lang        = \prop_item:cn { \l_tmpa_str } { lang } ,
3709         sig         = \prop_item:cn { \l_tmpa_str } { sig } ,
3710         meta        = \prop_item:cn { \l_tmpa_str } { meta } ,
3711         feature     = \prop_item:cn { \l_tmpa_str } { feature }
3712     }
3713 }
3714 } {
3715     \end{stex_annotate_env}
3716 }
3717 }
3718

```

32.3 Features

structure

```

3719
3720 \prop_new:N \l_stex_all_structures_prop
3721
3722 \keys_define:nn { stex / features / structure } {
3723     name          .str_set_x:N = \l__stex_features_structure_name_str ,
3724 }
3725
3726 \cs_new_protected:Nn \__stex_features_structure_args:n {
3727     \str_clear:N \l__stex_features_structure_name_str
3728     \keys_set:nn { stex / features / structure } { #1 }
3729 }
3730
3731 %\stex_new_feature:nnnn { structure } { 0{} m } {
3732 %   \__stex_features_structure_args:n { ##1 }
3733 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3734 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3735 %   }
3736 %} {
3737 %
3738 %}
3739
3740 \NewDocumentEnvironment{mathstructure}{0{} m }{
3741     \__stex_features_structure_args:n { #1 }
3742     \str_if_empty:NT \l__stex_features_structure_name_str {
3743         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3744     }
3745     \exp_args:Nnnx
3746     \begin{structural@feature}{ structure }
3747         { \l__stex_features_structure_name_str }{}
3748         \seq_clear:N \l_tmpa_seq
3749         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3750     \stex_smsmode_do:
3751 }{
3752     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3753     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3754     \str_set:Nx \l_tmpa_str {

```

```

3755     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3756     \prop_item:Nn \l_stex_current_module_prop { name }
3757   }
3758   \seq_map_inline:Nn \l_tmpa_seq {
3759     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3760   }
3761   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3762   \exp_args:Nnx
3763   \AddToHookNext { env / mathstructure / after }{
3764     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3765       \_stex_term_math_oms:nxxx { \l_tmpa_str }{}{0}{}
3766     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3767   \STEXexport {
3768     \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3769       {\prop_item:Nn \l_stex_current_module_prop { origname }}
3770       {\l_tmpa_str}
3771     \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3772       {#2}{\l_tmpa_str}
3773   %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3774   %     \prop_item:Nn \l_stex_current_module_prop { origname },
3775   %     \l_tmpa_str
3776   %   }
3777   %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3778   %     #2,\l_tmpa_str
3779   %   }
3780   %   \tl_set:cx { #2 } {
3781   %     \stex_invoke_structure:n { \l_tmpa_str }
3782   %   }
3783   }
3784
3785   \end{structural@feature}
3786   % \g_stex_last_feature_prop
3787 }

```

\instantiate

```

3788 \seq_new:N \l__stex_features_structure_field_seq
3789 \str_new:N \l__stex_features_structure_field_str
3790 \str_new:N \l__stex_features_structure_def_tl
3791 \prop_new:N \l__stex_features_structure_prop
3792 \NewDocumentCommand \instantiate { m O{} m }{
3793   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3794   \prop_set_eq:Nc \l__stex_features_structure_prop {
3795     c_stex_feature_\l_tmpa_str _prop
3796   }
3797   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3798   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3799     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3800     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3801       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3802       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3803         {!} \l_tmpa_tl
3804       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3805         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3806         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3807     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3808   }{
3809     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3810     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3811     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3812       \l_tmpa_tl
3813     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3814       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3815       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3816     }{
3817       \tl_clear:N \l_tmpb_tl
3818     }
3819   }
3820 }{
3821   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3822   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3823     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3824     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3825     \tl_clear:N \l_tmpa_tl
3826   }{
3827     % TODO throw error
3828   }
3829 }
3830 % \l_tmpa_str: name
3831 % \l_tmpa_tl: definiens
3832 % \l_tmpb_tl: notation
3833 \tl_if_empty:NT \l__stex_features_structure_field_str {
3834   % TODO throw error
3835 }
3836 \str_clear:N \l_tmpb_str
3837
3838 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3839 \seq_map_inline:Nn \l_tmpa_seq {
3840   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3841   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3842   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3843     \seq_map_break:n {
3844       \str_set:Nn \l_tmpb_str { ####1 }
3845     }
3846   }
3847 }
3848 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3849   \l_tmpb_str
3850
3851 \tl_if_empty:NNTF \l_tmpb_tl {
3852   \tl_if_empty:NF \l_tmpa_tl {
3853     \exp_args:Nx \use:n {
3854       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3855     }
3856   }
3857 }{
3858   \tl_if_empty:NNTF \l_tmpa_tl {
3859     \exp_args:Nx \use:n {
3860       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3861     }
3862
3863   }{
3864     \exp_args:Nx \use:n {
3865       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3866       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3867     }
3868   }
3869 }
3870 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3871 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3872 % #3/\l__stex_features_structure_field_str
3873 % \par
3874 % \expandafter\present\csname
3875 %   \l_stex_symdecl_
3876 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3877 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3878 %   #3/\l__stex_features_structure_field_str
3879 %   _prop
3880 % \endcsname
3881 }
3882
3883 \tl_clear:N \l__stex_features_structure_def_tl
3884
3885 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3886 \seq_map_inline:Nn \l_tmpa_seq {
3887   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3888   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3889   \exp_args:Nx \use:n {
3890     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3891
3892     }
3893   }
3894
3895   \prop_if_exist:cF {
3896     \l_stex_symdecl_
3897     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3898     \prop_item:Nn \l_stex_current_module_prop {name} ?
3899     #3/\l_tmpa_str
3900     _prop
3901   }{
3902     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3903     \l_tmpb_str
3904     \exp_args:Nx \use:n {
3905       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3906     }
3907   }
3908 }
3909
3910 \symdecl*[type={\STEXsymbol{module-type}}{
3911   \_stex_term_math_oms:nnnn {
3912     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3913     \prop_item:Nn \l__stex_features_structure_prop {name}
3914   }-{}{0}-{}

```

```

3915   }}]{#3}
3916
3917   % TODO: -> sms file
3918
3919   \tl_set:cx{ #3 }{
3920     \stex_invoke_structure:nnn {
3921       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3922       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3923     } {
3924       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3925       \prop_item:Nn \l__stex_features_structure_prop {name}
3926     }
3927   }
3928   \stex_smsmode_do:
3929 }

```

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

\stex_invoke_structure:nnn

```

3930 % #1: URI of the instance
3931 % #2: URI of the instantiated module
3932 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3933   \tl_if_empty:nTF{ #3 }{
3934     \prop_set_eq:Nc \l__stex_features_structure_prop {
3935       c_stex_feature_ #2 _prop
3936     }
3937     \tl_clear:N \l_tmpa_tl
3938     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3939     \seq_map_inline:Nn \l_tmpa_seq {
3940       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3941       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3942       \cs_if_exist:cT {
3943         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3944       }{
3945         \tl_if_empty:NF \l_tmpa_tl {
3946           \tl_put_right:Nn \l_tmpa_tl {,}
3947         }
3948         \tl_put_right:Nx \l_tmpa_tl {
3949           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3950         }
3951       }
3952     }
3953     \exp_args:No \mathstrut \l_tmpa_tl
3954   }{
3955     \stex_invoke_symbol:n{#1/#3}
3956   }
3957 }

```

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

3958 </package>

Chapter 33

STEX -Statements Implementation

```
3959 <*package>
3960
3961 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3962
3963 <@@=stex_statements>
    Warnings and error messages
3964
\titleemph
3965 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
3966 \keys_define:nn {stex / definiendum }{
3967   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3968   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3969   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3970 }
3971 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3972   \str_clear:N \l__stex_statements_definiendum_root_str
3973   \tl_clear:N \l__stex_statements_definiendum_post_tl
3974   \str_clear:N \l__stex_statements_definiendum_gfa_str
3975   \keys_set:nn { stex / definiendum }{ #1 }
3976 }
3977 \NewDocumentCommand \definiendum { 0{} m m } {
3978   \__stex_statements_definiendum_args:n { #1 }
3979   \stex_get_symbol:n { #2 }
3980   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3981   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3982     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3983       \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3984   } {
3985     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3986     \tl_set:Nn \l_tmpa_tl {
3987       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3988     }
3989   }
3990 } {
3991   \tl_set:Nn \l_tmpa_tl { #3 }
3992 }
3993
3994 % TODO root
3995 \rustex_if:TF {
3996   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3997 } {
3998   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3999 }
4000 }
4001 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4002
4003 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4004
4005 \NewDocumentCommand \definame { 0{ } m } {
4006   \__stex_statements_definiendum_args:n { #1 }
4007   % TODO: root
4008   \stex_get_symbol:n { #2 }
4009   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4010   \str_set:Nx \l_tmpa_str {
4011     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4012   }
4013   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4014   \rustex_if:TF {
4015     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4016       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4017     }
4018   } {
4019     \defemph@uri {
4020       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4021     } { \l_stex_get_symbol_uri_str }
4022   }
4023 }
4024 \stex_deactivate_macro:Nn \definame {definition~environments}
4025
4026 \NewDocumentCommand \Definame { 0{ } m } {
4027   \__stex_statements_definiendum_args:n { #1 }
4028   \stex_get_symbol:n { #2 }
4029   \str_set:Nx \l_tmpa_str {
4030     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4031   }
4032   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4033   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4034 \rustex_if:TF {
4035   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4036     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4037   }
4038 } {
4039   \defemph@uri {
4040     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4041   } { \l_stex_get_symbol_uri_str }
4042 }
4043 }
4044 \stex_deactivate_macro:Nn \Definame {definition-environments}
4045
4046 \NewDocumentCommand \Symname { 0{ } m }{
4047   \stex_symname_args:n { #1 }
4048   \stex_get_symbol:n { #2 }
4049   \str_set:Nx \l_tmpa_str {
4050     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4051   }
4052   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4053   \let\compemph_uri_prev:\compemph@uri
4054   \let\compemph@uri\symrefemph@uri
4055   \exp_args:NNx \use:nn
4056   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
4057     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4058     \l_stex_symname_post_str
4059   ] }
4060   \let\compemph@uri\compemph_uri_prev:
4061 }

```

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

sdefinition

```

4062
4063 \keys_define:nn {stex / sdefinition }{
4064   type      .str_set_x:N = \sdefinitiontype,
4065   id        .str_set_x:N = \sdefinitionid,
4066   name      .str_set_x:N = \sdefinitionname,
4067   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4068   title     .tl_set:N     = \sdefinitiontitle
4069 }
4070 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4071   \str_clear:N \sdefinitiontype
4072   \str_clear:N \sdefinitionid
4073   \str_clear:N \sdefinitionname
4074   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4075   \tl_clear:N \sdefinitiontitle
4076   \keys_set:nn { stex / sdefinition }{ #1 }
4077 }
4078
4079 \NewDocumentEnvironment{sdefinition}{0{}}{
4080   \__stex_statements_sdefinition_args:n{ #1 }
4081   \stex_reactivate_macro:N \definiendum
4082   \stex_reactivate_macro:N \definame
4083   \stex_reactivate_macro:N \Definame

```

```

4084 \stex_if_smsmode:F{
4085   \seq_clear:N \l_tmpa_seq
4086   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4087     \str_if_eq:nnF{ ##1 }{}{
4088       \stex_get_symbol:n { ##1 }
4089       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4090         \l_stex_get_symbol_uri_str
4091       }
4092     }
4093   }
4094   \exp_args:Nnnx
4095   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4096   \str_if_empty:NF \sdefinitiontype {
4097     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4098   }
4099   \clist_set:N \l_tmpa_clist \sdefinitiontype
4100   \tl_clear:N \l_tmpa_tl
4101   \clist_map_inline:Nn \l_tmpa_clist {
4102     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4103       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4104     }
4105   }
4106   \tl_if_empty:NTF \l_tmpa_tl {
4107     \__stex_statements_sdefinition_start:
4108   }{
4109     \l_tmpa_tl
4110   }
4111 }
4112 \stex_ref_new_doc_target:n \sdefinitionid
4113 \stex_smsmode_do:
4114 }{
4115   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4116   \stex_if_smsmode:F {
4117     \clist_set:N \l_tmpa_clist \sdefinitiontype
4118     \tl_clear:N \l_tmpa_tl
4119     \clist_map_inline:Nn \l_tmpa_clist {
4120       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4121         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4122       }
4123     }
4124     \tl_if_empty:NTF \l_tmpa_tl {
4125       \__stex_statements_sdefinition_end:
4126     }{
4127       \l_tmpa_tl
4128     }
4129     \end{stex_annotate_env}
4130   }
4131 }

```

\stexpatchdefinition

```

4132 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4133   \par\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
4134     ~(\sdefinitiontitle)
4135   }~}

```

```

4136 }
4137 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4138
4139 \newcommand\stexpatchdefinition[3] [] {
4140   \str_set:Nx \l_tmpa_str{ #1 }
4141   \str_if_empty:NTF \l_tmpa_str {
4142     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4143     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4144   }{
4145     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4146     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4147   }
4148 }

```

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

\inlinedef inline:

```

4149 \keys_define:nn {stex / inlinedef }{
4150   type      .str_set_x:N = \sdefinitiontype,
4151   id        .str_set_x:N = \sdefinitionid,
4152   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4153   name      .str_set_x:N = \sdefinitionname
4154 }
4155 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4156   \str_clear:N \sdefinitiontype
4157   \str_clear:N \sdefinitionid
4158   \str_clear:N \sdefinitionname
4159   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4160   \keys_set:nn { stex / inlinedef }{ #1 }
4161 }
4162 \NewDocumentCommand \inlinedef { 0{} m } {
4163   \begin{group}
4164     \__stex_statements_inlinedef_args:n{ #1 }
4165     \stex_ref_new_doc_target:n \sdefinitionid
4166     \stex_reactivate_macro:N \definiendum
4167     \stex_reactivate_macro:N \definame
4168     \stex_reactivate_macro:N \Definame
4169     \stex_if_smsmode:TF{
4170       \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4171     }{
4172       \seq_clear:N \l_tmpa_seq
4173       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4174         \str_if_eq:nnF{ ##1 }{ }{
4175           \stex_get_symbol:n { ##1 }
4176           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4177             \l_stex_get_symbol_uri_str
4178           }
4179         }
4180       }
4181       \exp_args:Nnx
4182       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4183         \str_if_empty:NF \sdefinitiontype {
4184           \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4185         }
4186       }

```

```

4186         #2
4187         \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4188     }
4189 }
4190 \endgroup
4191 \stex_smsmode_do:
4192 }

```

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

33.2 Assertions

sassertion

```

4193
4194 \keys_define:nn {stex / sassertion }{
4195     type      .str_set_x:N = \sassertiontype,
4196     id        .str_set_x:N = \sassertionid,
4197     title     .tl_set:N    = \sassertiontitle ,
4198     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4199     name      .str_set_x:N = \sassertionname
4200 }
4201 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
4202     \str_clear:N \sassertiontype
4203     \str_clear:N \sassertionid
4204     \str_clear:N \sassertionname
4205     \clist_clear:N \l__stex_statements_sassertion_for_clist
4206     \tl_clear:N \sassertiontitle
4207     \keys_set:nn { stex / sassertion }{ #1 }
4208 }
4209
4210 %\tl_new:N \g__stex_statements_aftergroup_tl
4211
4212 \NewDocumentEnvironment{sassertion}{0{}}{
4213     \l__stex_statements_sassertion_args:n{ #1 }
4214     \stex_if_smsmode:F {
4215         \seq_clear:N \l_tmpa_seq
4216         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4217             \str_if_eq:nnF{ ##1 }{}{
4218                 \stex_get_symbol:n { ##1 }
4219                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4220                     \l_stex_get_symbol_uri_str
4221                 }
4222             }
4223         }
4224         \exp_args:Nnnx
4225         \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4226         \str_if_empty:NF \sassertiontype {
4227             \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4228         }
4229         \clist_set:Nn \l_tmpa_clist \sassertiontype
4230         \tl_clear:N \l_tmpa_tl
4231         \clist_map_inline:Nn \l_tmpa_clist {
4232             \tl_if_exist:cT {\l__stex_statements_sassertion_##1_start:}{

```

```

4233     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4234   }
4235 }
4236 \tl_if_empty:NTF \l_tmpa_tl {
4237   \__stex_statements_sassertion_start:
4238 }{
4239   \l_tmpa_tl
4240 }
4241 }
4242 \stex_ref_new_doc_target:n \sassertionid
4243 \stex_smsmode_do:
4244 }{
4245   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4246   \stex_if_smsmode:F {
4247     \clist_set:No \l_tmpa_clist \sassertiontype
4248     \tl_clear:N \l_tmpa_tl
4249     \clist_map_inline:Nn \l_tmpa_clist {
4250       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4251         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4252       }
4253     }
4254     \tl_if_empty:NTF \l_tmpa_tl {
4255       \__stex_statements_sassertion_end:
4256     }{
4257       \l_tmpa_tl
4258     }
4259     \end{stex_annotate_env}
4260   }
4261 }

```

`\stexpatchassertion`

```

4262
4263 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4264   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4265     (\sassertiontitle)
4266   }~}
4267 }
4268 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4269
4270 \newcommand\stexpatchassertion[3] [] {
4271   \str_set:Nx \l_tmpa_str{ #1 }
4272   \str_if_empty:NTF \l_tmpa_str {
4273     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4274     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4275   }{
4276     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4277     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4278   }
4279 }

```

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

`\inlineass inline:`

```

4280 \keys_define:nn {stex / inlineass }{

```

```

4281 type      .str_set_x:N = \sassertiontype,
4282 id         .str_set_x:N = \sassertionid,
4283 for        .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4284 name       .str_set_x:N = \sassertionname
4285 }
4286 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4287   \str_clear:N \sassertiontype
4288   \str_clear:N \sassertionid
4289   \str_clear:N \sassertionname
4290   \clist_clear:N \l__stex_statements_sassertion_for_clist
4291   \keys_set:nn { stex / inlineass }{ #1 }
4292 }
4293 \NewDocumentCommand \inlineass { 0{} m } {
4294   \beginngroup
4295     \__stex_statements_inlineass_args:n{ #1 }
4296     \stex_ref_new_doc_target:n \sassertionid
4297     \stex_if_smsmode:TF{
4298       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4299     }{
4300       \seq_clear:N \l_tmpa_seq
4301       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4302         \str_if_eq:nnF{ ##1 }{}{
4303           \stex_get_symbol:n { ##1 }
4304           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4305             \l_stex_get_symbol_uri_str
4306           }
4307         }
4308       }
4309       \exp_args:Nnx
4310       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4311         \str_if_empty:NF \sassertiontype {
4312           \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4313         }
4314         #2
4315         \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4316       }
4317     }
4318   \endgroup
4319   \stex_smsmode_do:
4320 }

```

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

33.3 Examples

`sexample`

```

4321
4322 \keys_define:nn {stex / sexample }{
4323   type      .str_set_x:N = \exampletype,
4324   id        .str_set_x:N = \sexampleid,
4325   title     .tl_set:N     = \sexampletitle,
4326   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4327 }

```



```

4328 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4329   \str_clear:N \sexamplotype
4330   \str_clear:N \sexampleid
4331   \tl_clear:N \sexampltitle
4332   \clist_clear:N \l__stex_statements_sexample_for_clist
4333   \keys_set:nn { stex / sexample }{ #1 }
4334 }
4335
4336 \NewDocumentEnvironment{sexample}{0{}}{
4337   \__stex_statements_sexample_args:n{ #1 }
4338   \stex_if_smsmode:F {
4339     \seq_clear:N \l_tmpa_seq
4340     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4341       \str_if_eq:nnF{ ##1 }{{}{
4342         \stex_get_symbol:n { ##1 }
4343         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4344           \l_stex_get_symbol_uri_str
4345         }
4346       }
4347     }
4348     \exp_args:Nnnx
4349     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4350     \str_if_empty:NF \sexamplotype {
4351       \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4352     }
4353     \clist_set:No \l_tmpa_clist \sexamplotype
4354     \tl_clear:N \l_tmpa_tl
4355     \clist_map_inline:Nn \l_tmpa_clist {
4356       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4357         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4358       }
4359     }
4360     \tl_if_empty:NTF \l_tmpa_tl {
4361       \__stex_statements_sexample_start:
4362     }{
4363       \l_tmpa_tl
4364     }
4365   }
4366   \stex_ref_new_doc_target:n \sexampleid
4367   \stex_smsmode_do:
4368 }{
4369   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4370   \stex_if_smsmode:F {
4371     \clist_set:No \l_tmpa_clist \sexamplotype
4372     \tl_clear:N \l_tmpa_tl
4373     \clist_map_inline:Nn \l_tmpa_clist {
4374       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4375         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4376       }
4377     }
4378     \tl_if_empty:NTF \l_tmpa_tl {
4379       \__stex_statements_sexample_end:
4380     }{
4381       \l_tmpa_tl

```

```

4382     }
4383     \end{stex_annotate_env}
4384   }
4385 }

```

`\stexpatchexample`

```

4386
4387 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4388   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4389     (\sexampltitle)
4390   }~}
4391 }
4392 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4393
4394 \newcommand\stexpatchexample[3]{} {
4395   \str_set:Nx \l_tmpa_str{ #1 }
4396   \str_if_empty:NTF \l_tmpa_str {
4397     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4398     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4399   }{
4400     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4401     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4402   }
4403 }

```

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

`\inlineex` inline:

```

4404 \keys_define:nn {stex / inlineex }{
4405   type      .str_set_x:N = \sexampltype,
4406   id        .str_set_x:N = \sexampleid,
4407   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4408   name      .str_set_x:N = \sexamplname
4409 }
4410 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4411   \str_clear:N \sexampltype
4412   \str_clear:N \sexampleid
4413   \str_clear:N \sexamplname
4414   \clist_clear:N \l__stex_statements_sexample_for_clist
4415   \keys_set:nn { stex / inlineex }{ #1 }
4416 }
4417 \NewDocumentCommand \inlineex { 0{} m } {
4418   \begingroup
4419   \__stex_statements_inlineex_args:n{ #1 }
4420   \stex_ref_new_doc_target:n \sexampleid
4421   \stex_if_smsmode:TF{
4422     \str_if_empty:NF \sexamplname { \symdecl*{\examplename} }
4423   }{
4424     \seq_clear:N \l_tmpa_seq
4425     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4426       \str_if_eq:nnF{ ##1 }{}{
4427         \stex_get_symbol:n { ##1 }
4428         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4429           \l_stex_get_symbol_uri_str

```

```

4430     }
4431   }
4432 }
4433 \exp_args:Nnx
4434 \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4435   \str_if_empty:NF \sexamplotype {
4436     \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4437   }
4438   #2
4439   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4440 }
4441 }
4442 \endgroup
4443 \stex_smsmode_do:
4444 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4445 \keys_define:nn { stex / sparagraph } {
4446   id      .str_set_x:N = \sparagraphid ,
4447   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4448   type    .str_set_x:N = \sparagraphtype ,
4449   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
4450   from    .tl_set:N     = \sparagraphfrom ,
4451   to      .tl_set:N     = \sparagraphto ,
4452   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
4453   name    .str_set:N    = \sparagraphname
4454 }
4455
4456 \cs_new_protected:Nn \stex_sparagraph_args:n {
4457   \tl_clear:N \l_stex_sparagraph_title_tl
4458   \tl_clear:N \sparagraphfrom
4459   \tl_clear:N \sparagraphto
4460   \tl_clear:N \l_stex_sparagraph_start_tl
4461   \str_clear:N \sparagraphid
4462   \str_clear:N \sparagraphtype
4463   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4464   \str_clear:N \sparagraphname
4465   \keys_set:nn { stex / sparagraph } { #1 }
4466 }
4467 \newif\if@in@omtext\@in@omtextfalse
4468
4469 \NewDocumentEnvironment {sparagraph} { 0{ } } {
4470   \stex_sparagraph_args:n { #1 }
4471   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4472     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4473   }{
4474     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4475   }
4476   \@in@omtexttrue

```

```

4477 \stex_if_smsmode:F {
4478   \seq_clear:N \l_tmpa_seq
4479   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4480     \str_if_eq:nnF{ ##1 }{}{
4481       \stex_get_symbol:n { ##1 }
4482       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4483         \l_stex_get_symbol_uri_str
4484       }
4485     }
4486   }
4487   \exp_args:Nnnx
4488   \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {},}
4489   \str_if_empty:NF \sparagraphtype {
4490     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4491   }
4492   \str_if_empty:NF \sparagraphfrom {
4493     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4494   }
4495   \str_if_empty:NF \sparagraphto {
4496     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4497   }
4498   \clist_set:No \l_tmpa_clist \sparagraphtype
4499   \tl_clear:N \l_tmpa_tl
4500   \clist_map_inline:Nn \sparagraphtype {
4501     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4502       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4503     }
4504   }
4505   \tl_if_empty:NTF \l_tmpa_tl {
4506     \__stex_statements_sparagraph_start:
4507   }{
4508     \l_tmpa_tl
4509   }
4510 }
4511 \stex_ref_new_doc_target:n \sparagraphid
4512 \stex_smsmode_do:
4513 \ignorespacesandpars
4514 }{
4515   \stex_if_smsmode:F {
4516     \clist_set:No \l_tmpa_clist \sparagraphtype
4517     \tl_clear:N \l_tmpa_tl
4518     \clist_map_inline:Nn \l_tmpa_clist {
4519       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4520         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4521       }
4522     }
4523     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4524     \tl_if_empty:NTF \l_tmpa_tl {
4525       \__stex_statements_sparagraph_end:
4526     }{
4527       \l_tmpa_tl
4528     }
4529     \end{stex_annotate_env}
4530   }

```

4531 }

\stexpatchparagraph

4532

```
4533 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {  
4534   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {  
4535     \tl_if_empty:NF \l_stex_sparagraph_title_tl {  
4536       \titleemph{\l_stex_sparagraph_title_tl}:~  
4537     }  
4538   }{  
4539     \titleemph{\l_stex_sparagraph_start_tl}~  
4540   }  
4541 }
```

4538

```
4539   \titleemph{\l_stex_sparagraph_start_tl}~  
4540 }
```

4541

```
4542 \cs_new_protected:Nn \__stex_statements_sparagraph_end: { \par\medskip }
```

4543

```
4544 \newcommand\stexpatchparagraph[3] [] {
```

```
4545   \str_set:Nx \l_tmpa_str{ #1 }
```

```
4546   \str_if_empty:NTF \l_tmpa_str {
```

```
4547     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
```

```
4548     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
```

4549

```
4550     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
```

```
4551     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
```

4552

```
4553   }
```

4554

```
4555 \keys_define:nn { stex / inlinepara } {
```

```
4556   id      .str_set_x:N = \sparagraphid ,
```

```
4557   type    .str_set_x:N = \sparagraphtype ,
```

```
4558   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
```

```
4559   from    .tl_set:N     = \sparagraphfrom ,
```

```
4560   to      .tl_set:N     = \sparagraphto ,
```

```
4561   name    .str_set:N     = \sparagraphname
```

4562 }

```
4563 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
```

```
4564   \tl_clear:N \sparagraphfrom
```

```
4565   \tl_clear:N \sparagraphto
```

```
4566   \str_clear:N \sparagraphid
```

```
4567   \str_clear:N \sparagraphtype
```

```
4568   \clist_clear:N \l__stex_statements_sparagraph_for_clist
```

```
4569   \str_clear:N \sparagraphname
```

```
4570   \keys_set:nn { stex / inlinepara } { #1 }
```

4571 }

```
4572 \NewDocumentCommand \inlinepara { 0{} m } {
```

```
4573   \begingroup
```

```
4574   \__stex_statements_inlinepara_args:n{ #1 }
```

```
4575   \stex_ref_new_doc_target:n \sparagraphid
```

```
4576   \stex_if_smsmode:TF{
```

```
4577     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
```

4578

```
4579   }{  
4579     \seq_clear:N \l_tmpa_seq
```

```
4580     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
```

```
4581       \str_if_eq:nnF{ ##1 }{ }{
```

```
4582         \stex_get_symbol:n { ##1 }
```

```

4583     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4584         \l_stex_get_symbol_uri_str
4585     }
4586 }
4587 }
4588 \exp_args:Nnx
4589 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4590     \str_if_empty:NF \sparagraphtype {
4591         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4592     }
4593     \str_if_empty:NF \sparagraphfrom {
4594         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4595     }
4596     \str_if_empty:NF \sparagraphto {
4597         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4598     }
4599     #2
4600     \str_if_empty:NF \sparagraphname { \symdecl*{\sparagraphname} }
4601 }
4602 }
4603 \endgroup
4604 \stex_smsmode_do:
4605 }
4606

```

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

symboldoc

```

4607 \NewDocumentEnvironment{symboldoc}{m}{
4608     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4609     \seq_clear:N \l_tmpb_seq
4610     \seq_map_inline:Nn \l_tmpa_seq {
4611         \str_if_eq:nnF{ ##1 }{}{
4612             \stex_get_symbol:n { ##1 }
4613             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4614                 \l_stex_get_symbol_uri_str
4615             }
4616         }
4617     }
4618     \par
4619     \exp_args:Nnnx
4620     \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4621 }{
4622     \end{stex_annotate_env}
4623 }
4624 \</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

```
4625 <*package>
4626 <@@=stex_sproof>
4627
4628 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4629
```

34.2 Proofs

We first define some keys for the proof environment.

```
4630 \keys_define:nn { stex / spf } {
4631   id          .str_set:N = \l__stex_sproof_spf_id_str,
4632   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4633   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4634   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4635   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4636   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4637   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4638   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4639   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4640   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4641 }
4642 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4643   \str_clear:N \l__stex_sproof_spf_id_str
4644   \tl_clear:N \l__stex_sproof_spf_display_tl
4645   \tl_clear:N \l__stex_sproof_spf_for_tl
4646   \tl_clear:N \l__stex_sproof_spf_from_tl
4647   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4648   \tl_clear:N \l__stex_sproof_spf_type_tl
4649   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4650 \tl_clear:N \l__stex_sproof_spf_continues_tl
4651 \tl_clear:N \l__stex_sproof_spf_functions_tl
4652 \tl_clear:N \l__stex_sproof_spf_method_tl
4653 \keys_set:nn { stex / spf }{ #1 }
4654 }

```

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

```

4655 \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.

```

4656 \newcount\count_ten
4657 \newenvironment{pst@with@label}[1]{
4658   \edef\pst@label{#1}
4659   \advance\count_ten by 1\relax
4660   \count_ten=1
4661 }{
4662   \advance\count_ten by -1\relax
4663 }

```

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

```

4664 \def\the@pst@label{
4665   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4666 }

```

(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.

```

4667 \keys_define:nn { stex / pstlabel }{
4668   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4669   delimiter    .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4670   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4671 }
4672 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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


```

4673 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4674 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4675 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4676 }
4677 \__stex_sproof_pstlabel_args:n {}
4678 \newcommand\setpstlabelstyle[1]{
4679   \__stex_sproof_pstlabel_args:n {#1}
4680 }
4681 \newcommand\setpstlabelstyledefault{%
4682   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4683 }

```

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

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

```

4684 \ExplSyntaxOff
4685 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4686 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4687 \def\pst@make@label@short#1#2{#2}
4688 \def\pst@make@label@empty#1#2{}
4689 \ExplSyntaxOn
4690 \def\pstlabelstyle#1{%
4691   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4692 }%
4693 \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.

```

4694 \def\next@pst@label{%
4695   \global\advance\count\count10 by 1%
4696 }%

```

(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

```

4697 \def\sproof@box{
4698   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4699 }
4700 \def\spf@proofend{\sproof@box}
4701 \def\sproofend{
4702   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4703     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4704   }
4705 }
4706 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4707 \def\spf@proofsketch@kw{Proof Sketch}
4708 \def\spf@proof@kw{Proof}
4709 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4710 \AddToHook{begindocument}{
4711   \ltx@ifpackageloaded{babel}{
4712     \makeatletter
4713     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4714     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4715       \input{sproof-ngerman.ldf}
4716     }
4717     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4718       \input{sproof-finnish.ldf}
4719     }
4720     \clist_if_in:NnT \l_tmpa_clist {french}{
4721       \input{sproof-french.ldf}
4722     }
4723     \clist_if_in:NnT \l_tmpa_clist {russian}{
4724       \input{sproof-russian.ldf}
4725     }
4726     \makeatother
4727   }{}
4728 }

```

`spfsketch`

```

4729 \newcommand\spfsketch[2][]{
4730   \__stex_sproof_spf_args:n{#1}
4731   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4732     \titleemph{
4733       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4734         \spf@proofsketch@kw
4735       }{
4736         \l__stex_sproof_spf_type_tl
4737       }
4738     }:
4739   }
4740   {~#2}
4741   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4742   \sproofend
4743 }

```

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

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

```

4744 \newenvironment{spfeq}[2][]{
4745   \__stex_sproof_spf_args:n{#1}
4746   %\sref@target
4747   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4748     \titleemph{
4749       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
4750         \spf@proof@kw
4751       }{

```

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

¹⁵EDNOTE: document above

```

4752         \l__stex_sproof_spf_type_tl
4753     }
4754     }:
4755 }
4756 {-#2}
4757 \begin{displaymath}\begin{array}{rcll}
4758 }{
4759 \end{array}\end{displaymath}
4760 }

```

(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.

```

4761 \newenvironment{spf@proof}[2] []{
4762   \l__stex_sproof_spf_args:n{#1}
4763   %\sref@target
4764   \count_ten=10
4765   \par\noindent
4766   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4767     \titleemph{
4768       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4769         \spf@proof@kw
4770       }{
4771         \l__stex_sproof_spf_type_tl
4772       }
4773     }:
4774   }
4775   {-#2}
4776   %\sref@label{id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}}
4777   \def\pst@label{}
4778   \newcount\pst@count% initialize the labeling mechanism
4779   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4780   }{
4781     \end{pst@with@label}\end{description}
4782   }
4783   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4784   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4785 \newcommand\spfidea[2] []{
4786   \l__stex_sproof_spf_args:n{#1}
4787   \titleemph{
4788     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4789       \l__stex_sproof_spf_type_tl
4790     }:
4791   }-#2
4792   \sproofend
4793 }

```

(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
4794 \newenvironment{spfstep}[1][]{
4795   \_stex_sproof_spf_args:n{#1}
4796   \@in@omtexttrue
4797   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4798     \item[\the@pst@label]
4799   }
4800   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4801     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4802   }
4803   %\sref@label{id{\pst@label}
4804   \ignorespacesandpars
4805 }{
4806   \next@pst@label\ignorespacesandpars
4807 }

```

sproofcomment

```

4808 \newenvironment{sproofcomment}[1][]{
4809   \_stex_sproof_spf_args:n{#1}
4810   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4811     \item[\the@pst@label]
4812   }
4813 }{
4814   \next@pst@label
4815 }

```

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.

```

4816 \newenvironment{subproof}[2][]{
4817   \_stex_sproof_spf_args:n{#1}
4818   \def\@test{#2}
4819   \ifx\@test\empty\else
4820     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4821       \item[\the@pst@label]
4822     }{#2}
4823   \fi
4824   \begin{pst@with@label}{\pst@label,\number\count_ten}
4825 }{
4826   \end{pst@with@label}\next@pst@label
4827 }

```

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

```

4828 \newenvironment{spfcases}[2][]{
4829   \def\@test{#1}
4830   \ifx\@test\empty
4831     \begin{subproof}[method=by-cases]{#2}

```

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

```

4832 \else
4833   \begin{subproof}[#1,method=by-cases]{#2}
4834 \fi
4835 }{
4836   \end{subproof}
4837 }

```

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

```

4838 \newenvironment{spfcase}[2] [] {
4839   \__stex_sproof_spf_args:n{#1}
4840   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4841     \item[\the@pst@label]
4842   }
4843   \def\@test{#2}
4844   \ifx\@test\@empty
4845   \else
4846     {\titleemph{#2}:~}
4847   \fi
4848   \begin{pst@with@label}{\pst@label,\number\count_ten}
4849 }{
4850   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4851     \sproofend
4852   }
4853   \end{pst@with@label}
4854   \next@pst@label
4855 }

```

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

```

4856 \newcommand\spfcasesketch[3] [] {
4857   \__stex_sproof_spf_args:n{#1}
4858   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4859     \item[\the@pst@label]
4860   }
4861   \def\@test{#2}
4862   \ifx\@test\@empty
4863   \else
4864     {\titleemph{#2}:~}
4865   \fi#3
4866   \next@pst@label
4867 }%

```

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.

```

4868 \keys_define:nn { stex / just }{
4869   id      .str_set:x:N = \l__stex_sproof_just_id_str,
4870   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
4871   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
4872   args    .tl_set:N    = \l__stex_sproof_just_args_tl
4873 }

```

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

`justification`

`4874 \newenvironment{justification}[1] [] {}{}`

`\premise`

`4875 \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.

`4876 \newcommand\justarg[2] [] {#2}`

`4877 \end{package}`

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

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

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

Chapter 35

STEX -Others Implementation

```
4878 <*package>
4879
4880 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4881
4882 <@@=stex_others>
    Warnings and error messages
4883 % None

\MSC Math subject classifier

4884 \NewDocumentCommand \MSC {m} {
4885 % TODO
4886 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4887 \@ifpackageloaded{tikzinput}{
4888 \RequirePackage{stex-tikzinput}
4889 }{}
4890 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4891 \*package>
4892 \@@=stex_modules>
4893
4894 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4895
4896 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4897 \begingroup
4898 \stex_module_setup:nn{
4899   ns=\c_stex_metatheory_ns_str,
4900   meta=NONE
4901 }{Metatheory}
4902 \stex_reactivate_macro:N \symdecl
4903 \stex_reactivate_macro:N \notation
4904 \stex_reactivate_macro:N \symdef
4905 \ExplSyntaxOff
4906 \csname stex_suppress_html:n\endcsname{
4907   % is-a (a:A, a \in A, a is an A, etc.)
4908   \symdecl[args=ai]{isa}
4909   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4910   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4911   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4912
4913   % bind (\forall, \Pi, \lambda etc.)
4914   \symdecl[args=Bi]{bind}
4915   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4916   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4917   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4918
4919   % dummy variable
4920   \symdecl{dummyvar}
4921   \notation[underscore]{dummyvar}{\comp\_}
4922   \notation[dot]{dummyvar}{\comp\cdot}
4923   \notation[dash]{dummyvar}{\comp{\rm --}}
4924
4925   %fromto (function space, Hom-set, implication etc.)
```



```

4926 \symdecl[args=ai]{fromto}
4927 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4928 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4929
4930 % mapto (lambda etc.)
4931 \%symdecl[args=Bi]{mapto}
4932 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4933 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4934 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4935
4936 % function/operator application
4937 \symdecl[args=ia]{apply}
4938 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4939 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4940
4941 % ‘type’ of all collections (sets, classes, types, kinds)
4942 \symdecl{collection}
4943 \notation[U]{collection}{\comp{\mathcal{U}}}
4944 \notation[set]{collection}{\comp{\textsf{Set}}}
4945
4946 % sequences
4947 \symdecl[args=1]{seqtype}
4948 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4949
4950 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4951 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{\#2}
4952
4953 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4954 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4955 \symdef[args=aia,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4956
4957 % letin (‘let’, local definitions, variable substitution)
4958 \symdecl[args=bii]{letin}
4959 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4960 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4961 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4962
4963 % structures
4964 \symdecl*[args=1]{module-type}
4965 \notation{module-type}{\mathtt{MOD} #1}
4966 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4967 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4968
4969 }
4970 \ExplSyntaxOn
4971 \stex_add_to_current_module:n{
4972   \let\nappa\apply
4973   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4974   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4975   \def\livar{\csname sequence-index\endcsname[li]}
4976   \def\uivar{\csname sequence-index\endcsname[ui]}
4977   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4978   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4979   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```

```
4980 }  
4981 \_stex_modules_end_module:  
4982 \endgroup  
4983 </package>
```

Chapter 37

Tikzinput Implementation

```
4984 <*package>
4985
4986 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4987
4988 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4989 \RequirePackage{l3keys2e}
4990
4991 \keys_define:nn { tikzinput } {
4992   image .bool_set:N = \c_tikzinput_image_bool,
4993   image .default:n = false ,
4994   unknown .code:n = {}
4995 }
4996
4997 \ProcessKeysOptions { tikzinput }
4998
4999 \bool_if:NTF \c_tikzinput_image_bool {
5000   \RequirePackage{graphicx}
5001
5002   \providecommand\usetikzlibrary[]{}
5003   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5004 }{
5005   \RequirePackage{tikz}
5006   \RequirePackage{standalone}
5007
5008   \newcommand \tikzinput [2] [] {
5009     \setkeys{Gin}{#1}
5010     \ifx \Gin@ewidth \Gin@exclamation
5011       \ifx \Gin@eheight \Gin@exclamation
5012         \input { #2 }
5013       \else
5014         \resizebox{!}{ \Gin@eheight }{
5015           \input { #2 }
5016         }
5017       \fi
5018     \else
5019       \ifx \Gin@eheight \Gin@exclamation
5020         \resizebox{ \Gin@ewidth }{!}{
5021           \input { #2 }
```

```

5022     }
5023     \else
5024         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5025             \input { #2 }
5026         }
5027     \fi
5028 \fi
5029 }
5030 }
5031
5032 \newcommand \ctikzinput [2] [] {
5033     \begin{center}
5034         \tikzinput [1] {#2}
5035     \end{center}
5036 }
5037
5038 \@ifpackageloaded{stex}{
5039     \RequirePackage{stex-tikzinput}
5040 }{}
5041
5042 </package>
5043 <*stex>
5044 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5045 \RequirePackage{stex}
5046 \RequirePackage{tikzinput}
5047
5048 \newcommand\mhtikzinput [2] [] {%
5049     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5050     \stex_in_repository:nn\Gin@mhrepos{
5051         \tikzinput[#1]{\mhpath{##1}{#2}}
5052     }
5053 }
5054 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5055 </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.

```
5056 \*cls)
5057 \<@@=document_structure>
5058 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5059 \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`

```
5060 \keys_define:nn{ document-structure / pkg }{
5061   class      .str_set_x:N = \c_document_structure_class_str,
5062   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5063   report     .code:n      = {
5064     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5065     \str_set:Nn \c_document_structure_class_str {report}
5066   },
5067   book       .code:n      = {
5068     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5069     \str_set:Nn \c_document_structure_class_str {book}
5070   },
5071   bookpart   .code:n      = {
5072     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5073     \str_set:Nn \c_document_structure_class_str {book}
5074     \str_set:Nn \c_document_structure_topsect_str {chapter}
5075   },
```

```

5076 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5077 unknown     .code:n      = {
5078   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5079 }
5080 }
5081 \ProcessKeysOptions{ document-structure / pkg }
5082 \str_if_empty:NT \c_document_structure_class_str {
5083   \str_set:Nn \c_document_structure_class_str {article}
5084 }
5085 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5086   {\c_document_structure_class_str}
5087

```

38.3 Beefing up the document environment

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

```

5088 \RequirePackage{document-structure}
5089 \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

```

5090 \keys_define:nn { document-structure / document }{
5091   id .str_set_x:N = \c_document_structure_document_id_str
5092 }
5093 \let\__document_structure_orig_document=\document
5094 \renewcommand{\document}[1][]{
5095   \keys_set:nn{ document-structure / document }{ #1 }
5096   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5097   \__document_structure_orig_document
5098 }

```

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

```

5099 }
5100 </cls>

```

38.4 Implementation: document-structure Package

```

5101 <*package>
5102 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5103 \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

```

5104
5105 \keys_define:nn{ document-structure / pkg }{
5106   class      .str_set_x:N = \c_document_structure_class_str,
5107   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5108   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5109 }
5110 \ProcessKeysOptions{ document-structure / pkg }
5111 \str_if_empty:NT \c_document_structure_class_str {
5112   \str_set:Nn \c_document_structure_class_str {article}
5113 }
5114 \str_if_empty:NT \c_document_structure_topsect_str {
5115   \str_set:Nn \c_document_structure_topsect_str {section}
5116 }

```

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

```

5117 \RequirePackage{xspace}
5118 \RequirePackage{comment}
5119 \AddToHook{begindocument}{
5120   \ltx@ifpackageloaded{babel}{
5121     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5122     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5123       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5124     }
5125   }{}
5126 }

```

`\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.

```

5127 \int_new:N \l_document_structure_section_level_int
5128 \str_case:VnF \c_document_structure_topsect_str {
5129   {part}}{
5130     \int_set:Nn \l_document_structure_section_level_int {0}
5131   }
5132   {chapter}{
5133     \int_set:Nn \l_document_structure_section_level_int {1}
5134   }
5135 }{
5136   \str_case:VnF \c_document_structure_class_str {
5137     {book}}{
5138       \int_set:Nn \l_document_structure_section_level_int {0}
5139     }
5140     {report}}{
5141       \int_set:Nn \l_document_structure_section_level_int {0}
5142     }
5143   }{
5144     \int_set:Nn \l_document_structure_section_level_int {2}
5145   }
5146 }

```

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

```
5147 \def\current@section@level{document}%
5148 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5149 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5150 \cs_new_protected:Npn \skipomgroup {
5151   \ifcase\l_document_structure_section_level_int
5152   \or\stepcounter{part}
5153   \or\stepcounter{chapter}
5154   \or\stepcounter{section}
5155   \or\stepcounter{subsection}
5156   \or\stepcounter{subsubsection}
5157   \or\stepcounter{paragraph}
5158   \or\stepcounter{subparagraph}
5159   \fi
5160 }
```

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

`blindomgroup`

```
5161 \newcommand\at@begin@blindomgroup[1]{%
5162 \newenvironment{blindomgroup}
5163 {
5164   \int_incr:N\l_document_structure_section_level_int
5165   \at@begin@blindomgroup\l_document_structure_section_level_int
5166 }{}}
```

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

```
5167 \newcommand\omgroup@nonum[2]{%
5168   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5169   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5170 }
```

(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.

```
5171 \newcommand\omgroup@num[2]{%
```

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


```

5172 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5173   \@nameuse{#1}{#2}
5174 }{
5175   \cs_if_exist:NTF\rdfmata@sectioning{
5176     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5177   }{
5178     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5179   }
5180 }
5181 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5182 }

```

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

omgroup

```

5183 \keys_define:nn { document-structure / omgroup }{
5184   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5185   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5186   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5187   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5188   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5189   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5190   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5191   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5192   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5193   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5194 }
5195 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5196   \str_clear:N \l__document_structure_omgroup_id_str
5197   \str_clear:N \l__document_structure_omgroup_date_str
5198   \clist_clear:N \l__document_structure_omgroup_creators_clist
5199   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5200   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5201   \tl_clear:N \l__document_structure_omgroup_type_tl
5202   \tl_clear:N \l__document_structure_omgroup_short_tl
5203   \tl_clear:N \l__document_structure_omgroup_display_tl
5204   \tl_clear:N \l__document_structure_omgroup_intro_tl
5205   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5206   \keys_set:nn { document-structure / omgroup } { #1 }
5207 }

```

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.

```

5208 \newif\if@mainmatter\@mainmattertrue
5209 \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.

```

5210 \keys_define:nn { document-structure / sectioning }{
5211   name .str_set_x:N = \l__document_structure_sect_name_str ,
5212   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5213   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5214   clear .default:n = {true} ,
5215   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5216   num      .default:n    = {true}
5217 }
5218 \cs_new_protected:Nn \__document_structure_sect_args:n {
5219   \str_clear:N \l__document_structure_sect_name_str
5220   \str_clear:N \l__document_structure_sect_ref_str
5221   \bool_set_false:N \l__document_structure_sect_clear_bool
5222   \bool_set_false:N \l__document_structure_sect_num_bool
5223   \keys_set:nn { document-structure / sectioning } { #1 }
5224 }
5225 \newcommand\omdoc@sectioning[3][]{
5226   \__document_structure_sect_args:n {#1}
5227   \let\omdoc@ssect@name\l__document_structure_sect_name_str
5228   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5229   \if@mainmatter% numbering not overridden by frontmatter, etc.
5230     \bool_if:NTF \l__document_structure_sect_num_bool {
5231       \omgroup@num{#2}{#3}
5232     }{
5233       \omgroup@nonum{#2}{#3}
5234     }
5235     \def\current@section@level{\omdoc@ssect@name}
5236   \else
5237     \omgroup@nonum{#2}{#3}
5238   \fi
5239 }% 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.

```

5240 \newcommand\omgroup@redefine@addtocontents[1]{%
5241 %\edef\__document_structureimport{#1}%
5242 %\@for\@I:=\__document_structureimport\do{%
5243 %\edef\@path{\csname module@\@I @path\endcsname}%
5244 %\@ifundefined{tf@toc}\relax%
5245 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5246 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5247 %\def\addcontentsline##1##2##3{%
5248 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5249 %\else% hyperref.sty not loaded
5250 %\def\addcontentsline##1##2##3{%
5251 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5252 %\fi
5253 }% hyperref.sty loaded?

```

now the `omgroup` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of `omgroups` in the `\omgroup@level` counter.

```

5254 \newenvironment{omgroup}[2][]{% keys, title
5255 {
5256   \__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.

```

5257 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5258   \omgroup@redefine@addtocontents{
5259     \@ifundefined{module@id}\used@modules%

```

```

5260     %{\ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5261   }
5262 }

```

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

```

5263 \int_incr:N\l_document_structure_section_level_int
5264 \ifcase\l_document_structure_section_level_int
5265   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5266   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5267   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5268   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5269   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5270   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5271   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5272 \fi
5273 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5274 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5275 }% for customization
5276 {}

```

and finally, we localize the sections

```

5277 \newcommand\omdoc@part@kw{Part}
5278 \newcommand\omdoc@chapter@kw{Chapter}
5279 \newcommand\omdoc@section@kw{Section}
5280 \newcommand\omdoc@subsection@kw{Subsection}
5281 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5282 \newcommand\omdoc@paragraph@kw{paragraph}
5283 \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`

```

5284 \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).

```

5285 \cs_if_exist:NTF\frontmatter{
5286   \let\__document_structure_orig_frontmatter\frontmatter
5287   \let\frontmatter\relax
5288 }{
5289   \tl_set:Nn\__document_structure_orig_frontmatter{
5290     \clearpage
5291     \@mainmatterfalse
5292     \pagenumbering{roman}
5293   }
5294 }

```

```

5295 \cs_if_exist:NTF\backmatter{
5296   \let\__document_structure_orig_backmatter\backmatter
5297   \let\backmatter\relax
5298 }{
5299   \tl_set:Nn\__document_structure_orig_backmatter{
5300     \clearpage
5301     \@mainmatterfalse
5302     \pagenumbering{roman}
5303   }
5304 }

```

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.

```

5305 \newenvironment{frontmatter}{
5306   \_document_structure_orig_frontmatter
5307 }{
5308   \cs_if_exist:NTF\mainmatter{
5309     \mainmatter
5310   }{
5311     \clearpage
5312     \@mainmattertrue
5313     \pagenumbering{arabic}
5314   }
5315 }

```

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

```

5316 \newenvironment{backmatter}{
5317   \__document_structure_orig_backmatter
5318 }{
5319   \cs_if_exist:NTF\mainmatter{
5320     \mainmatter
5321   }{
5322     \clearpage
5323     \@mainmattertrue
5324     \pagenumbering{arabic}
5325   }
5326 }

```

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

5327 \@mainmattertrue\pagenumbering{arabic}

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

```

5328 \def \c__document_structure_document_str{document}
5329 \newcommand\afterprematurestop{}
5330 \def\prematurestop@endomgroup{
5331   \unless\ifx\@currentvir\c__document_structure_document_str
5332     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5333     \expandafter\prematurestop@endomgroup
5334   \fi
5335 }

```

```

5336 \providecommand\prematurestop{
5337   \message{Stopping~sTeX~processing~prematurely}
5338   \prematurestop@endomgroup
5339   \afterprematurestop
5340   \end{document}}
5341 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5342 \RequirePackage{etoolbox}
5343 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5344 \newrobustcmd\useSGvar[1]{%
5345   \@ifundefined{sTeX@Gvar@#1}
5346   {\PackageError{document-structure}
5347    {The sTeX Global variable #1 is undefined}
5348    {set it with \protect\setSGvar}}
5349   \@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.

```

5350 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5351   \@ifundefined{sTeX@Gvar@#1}
5352   {\PackageError{document-structure}
5353    {The sTeX Global variable #1 is undefined}
5354    {set it with \protect\setSGvar}}
5355   {\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.

```
5356 \*cls)
5357 \@@=notesslides)
5358 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5359 \RequirePackage{l3keys2e,expl-keystr-compatible}
5360
5361 \keys_define:nn{notesslides / cls}{
5362   class .code:n = {
5363     \PassOptionsToClass{\CurrentOption}{document-structure}
5364     \str_if_eq:nnT{#1}{book}{
5365       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5366     }
5367     \str_if_eq:nnT{#1}{report}{
5368       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5369     }
5370   },
5371   notes .bool_set:N = \c__notesslides_notes_bool ,
5372   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5373   unknown .code:n = {
5374     \PassOptionsToClass{\CurrentOption}{document-structure}
5375     \PassOptionsToClass{\CurrentOption}{beamer}
5376     \PassOptionsToPackage{\CurrentOption}{notesslides}
5377   }
5378 }
5379 \ProcessKeysOptions{ notesslides / cls }
5380 \bool_if:NTF \c__notesslides_notes_bool {
5381   \PassOptionsToPackage{notes=true}{notesslides}
5382 }{
5383   \PassOptionsToPackage{notes=false}{notesslides}
5384 }
5385 \</cls)
```

now we do the same for the notesslides package.

```

5386 <*package>
5387 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5388 \RequirePackage{13keys2e,expl-keystr-compat}
5389
5390 \keys_define:nn{notesslides / pkg}{
5391   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5392   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5393   notes        .bool_set:N = \c__notesslides_notes_bool ,
5394   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5395   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5396   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5397   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5398   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5399   unknown      .code:n      = {
5400     \PassOptionsToClass{\CurrentOption}{stex}
5401     \PassOptionsToClass{\CurrentOption}{tikzinput}
5402   }
5403 }
5404 \ProcessKeysOptions{ notesslides / pkg }
5405 \newif\ifnotes
5406 \bool_if:NTF \c__notesslides_notes_bool {
5407   \notesttrue
5408 }{
5409   \notesfalse
5410 }
5411

```

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

```

5412 \str_if_empty:NTF \c__notesslides_topsect_str {
5413   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5414 }{
5415   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5416 }
5417 </package>

```

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

```

5418 <*cls>
5419 \bool_if:NTF \c__notesslides_notes_bool {
5420   \LoadClass{document-structure}
5421 }{
5422   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5423   \newcounter{Item}
5424   \newcounter{paragraph}
5425   \newcounter{subparagraph}
5426   \newcounter{Hfootnote}
5427   \RequirePackage{document-structure}
5428 }

```

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

```

5429 \RequirePackage{notesslides}
5430 </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.

```

5431 \*package>
5432 \bool_if:NT \c__notesslides_notes_bool {
5433   \RequirePackage{a4wide}
5434   \RequirePackage{marginnote}
5435   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5436   \RequirePackage{mdframed}
5437   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5438   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5439 }
5440 \RequirePackage{stex-tikzinput}
5441 \RequirePackage{etoolbox}
5442 \RequirePackage{amssymb}
5443 \RequirePackage{amsmath}
5444 \RequirePackage{comment}
5445 \RequirePackage{textcomp}
5446 \RequirePackage{url}
5447 \RequirePackage{graphicx}
5448 \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

```

5449 \bool_if:NT \c__notesslides_notes_bool {
5450   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5451 }

```

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

```

5452 \newcounter{slide}
5453 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5454 \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.

```

5455 \bool_if:NTF \c__notesslides_notes_bool {
5456   \renewenvironment{note}{\ignorespaces}{}
5457 }{
5458   \excludcomment{note}
5459 }

```

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.

```
5460 \bool_if:NT \c__notesslides_notes_bool {
5461   \newlength{\slideframewidth}
5462   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5463 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5464   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5465     \bool_set_true:N #1
5466   }{
5467     \bool_set_false:N #1
5468   }
5469 }
5470 \keys_define:nn{notesslides / frame}{
5471   label .str_set_x:N = \l__notesslides_frame_label_str,
5472   allowframebreaks .code:n = {
5473     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5474   },
5475   allowdisplaybreaks .code:n = {
5476     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5477   },
5478   fragile .code:n = {
5479     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5480   },
5481   shrink .code:n = {
5482     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5483   },
5484   squeeze .code:n = {
5485     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5486   },
5487   t .code:n = {
5488     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5489   },
5490 }
5491 \cs_new_protected:Nn \__notesslides_frame_args:n {
5492   \str_clear:N \l__notesslides_frame_label_str
5493   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5494   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5495   \bool_set_true:N \l__notesslides_frame_fragile_bool
5496   \bool_set_true:N \l__notesslides_frame_shrink_bool
5497   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5498   \bool_set_true:N \l__notesslides_frame_t_bool
5499   \keys_set:nn { notesslides / frame }{ #1 }
5500 }
```

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

```
5501 \renewenvironment{frame}[1][]{
5502   \__notesslides_frame_args:n{#1}
5503   \sffamily
5504   \stepcounter{slide}
5505   \def\@currentlabel{\theslide}
5506   \str_if_empty:NF \l__notesslides_frame_label_str {
5507     \label{\l__notesslides_frame_label_str}
```

5508 }
5509

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

5509 \def\itemize@level{outer}
5510 \def\itemize@outer{outer}
5511 \def\itemize@inner{inner}
5512 \renewcommand\newpage{\addtocounter{framenum}{1}}
5513 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5514 \renewenvironment{itemize}{
5515 \ifx\itemize@level\itemize@outer
5516 \def\itemize@label{\\$ \rhd\$}
5517 \fi
5518 \ifx\itemize@level\itemize@inner
5519 \def\itemize@label{\\$ \scriptstyle \rhd\$}
5520 \fi
5521 \begin{list}
5522 {\itemize@label}
5523 {\setlength{\labelsep}{.3em}
5524 \setlength{\labelwidth}{.5em}
5525 \setlength{\leftmargin}{1.5em}
5526 }
5527 \edef\itemize@level{\itemize@inner}
5528 }{
5529 \end{list}
5530 }

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

5531 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5532 }{
5533 \medskip\miko@slidelabel\end{mdframed}
5534 }

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

\frametitle

5535 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5536 }

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

EdN:21

\pause 21

5537 \bool_if:NT \c__notesslides_notes_bool {
5538 \newcommand\pause{
5539 }
5540 }

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

nparagraph

5540 \bool_if:NTF \c__notesslides_notes_bool {
5541 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5542 }{
5543 \excludecomment{nparagraph}
5544 }
5545 }

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

nomgroup

```
5545 \bool_if:NTF \c__notesslides_notes_bool {
5546   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5547 }{
5548   \excludecomment{nomgroup}
5549 }
```

ndefinition

```
5550 \bool_if:NTF \c__notesslides_notes_bool {
5551   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5552 }{
5553   \excludecomment{ndefinition}
5554 }
```

nassertion

```
5555 \bool_if:NTF \c__notesslides_notes_bool {
5556   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5557 }{
5558   \excludecomment{nassertion}
5559 }
```

nsproof

```
5560 \bool_if:NTF \c__notesslides_notes_bool {
5561   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5562 }{
5563   \excludecomment{nproof}
5564 }
```

nexample

```
5565 \bool_if:NTF \c__notesslides_notes_bool {
5566   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5567 }{
5568   \excludecomment{nexample}
5569 }
```

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

```
5570 \def\inputref@preskip{\smallskip}
5571 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5572 \let\orig@inputref\inputref
5573 \def\inputref{\@ifstar\ninputref\orig@inputref}
5574 \newcommand\ninputref[2] [] {
5575   \bool_if:NT \c__notesslides_notes_bool {
5576     \orig@inputref[#1]{#2}
5577   }
5578 }
```

(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>}`.

```

5579 \newlength{\slidelogoheight}
5580
5581 \bool_if:NTF \c__notesslides_notes_bool {
5582   \setlength{\slidelogoheight}{.4cm}
5583 }{
5584   \setlength{\slidelogoheight}{1cm}
5585 }
5586 \newsavebox{\slidelogo}
5587 \sbox{\slidelogo}{\TeX}
5588 \newrobustcmd{\setslidelogo}[1]{
5589   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5590 }
```

(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.

```

5591 \def\source{Michael Kohlhase}% customize locally
5592 \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.

```

5593 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5594 \newsavebox{\cclogo}
5595 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5596 \newif\ifcchref\cchreffalse
5597 \AtBeginDocument{
5598   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5599 }
5600 \def\licensing{
5601   \ifcchref
5602     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5603   \else
5604     {\usebox{\cclogo}}
5605   \fi
5606 }
5607 \newrobustcmd{\setlicensing}[2][]{
5608   \def@url{#1}
5609   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5610   \ifx@url@empty
5611     \def\licensing{{\usebox{\cclogo}}}
5612   \else
5613     \def\licensing{
```

```

5614     \ifcchref
5615     \href{#1}{\usebox{\cclogo}}
5616   \else
5617     {\usebox{\cclogo}}
5618   \fi
5619 }
5620 \fi
5621 }

```

(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

```

5622 \newrobustcmd\miko@slidelabel{
5623   \vbox to \slidelogoheight{
5624     \vss\hbox to \slidewidth
5625     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5626   }
5627 }

```

(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.

```

5628 \def\Gin@mhrepos{}
5629 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5630 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5631 \newrobustcmd\frameimage[2][]{
5632   \stepcounter{slide}
5633   \bool_if:NT \c__notesslides_frameimages_bool {
5634     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5635     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5636     \begin{center}
5637       \bool_if:NTF \c__notesslides_fiboxed_bool {
5638         \fbox{
5639           \ifx\Gin@ewidth\@empty
5640             \ifx\Gin@mhrepos\@empty
5641               \mhgraphics[width=\slidewidth,#1]{#2}
5642             \else
5643               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5644             \fi
5645           \else% Gin@ewidth empty
5646             \ifx\Gin@mhrepos\@empty
5647               \mhgraphics[#1]{#2}
5648             \else
5649               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5650             \fi
5651           \fi% Gin@ewidth empty
5652         }
5653       }{
5654         \ifx\Gin@ewidth\@empty

```

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

```

5655         \ifx\Gin@mhrepos\@empty
5656             \mhgraphics[width=\slidewidth,#1]{#2}
5657         \else
5658             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5659         \fi
5660         \ifx\Gin@mhrepos\@empty
5661             \mhgraphics[#1]{#2}
5662         \else
5663             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5664         \fi
5665     \fi% Gin@ewidth empty
5666 }
5667 \end{center}
5668 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5669 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5670 }
5671 } % 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.

```

5672 \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

```

5673 \AddToHook{begindocument}{
5674     \definecolor{green}{rgb}{0,.5,0}
5675     \definecolor{purple}{cmyk}{.3,1,0,.17}
5676 }

```

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`.

```

5677 % \def\STpresent#1{\textcolor{blue}{#1}}
5678 \def\defemph#1{\textcolor{magenta}{#1}}
5679 \def\symrefemph#1{\textcolor{cyan}{#1}}
5680 \def\compemph#1{\textcolor{blue}{#1}}
5681 \def\titleemph#1{\textcolor{blue}{#1}}
5682 \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.

```

5683 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5684 \def\smalltextwarning{
5685     \pgfuseimage{miko@small@dbend}
5686     \xspace
5687 }
5688 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5689 \newrobustcmd\textwarning{
5690   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5691   \xspace
5692 }
5693 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5694 \newrobustcmd\bigtextwarning{
5695   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5696   \xspace
5697 }

```

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

```

5698 \newrobustcmd\putgraphicsat[3]{
5699   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5700 }
5701 \newrobustcmd\putat[2]{
5702   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5703 }

```

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`.

```

5704 \bool_if:NT \c__notesslides_sectocframes_bool {
5705   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5706     \newcounter{chapter}\counterwithin*{section}{chapter}
5707   }{
5708     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5709       \newcounter{chapter}\counterwithin*{section}{chapter}
5710     }
5711   }
5712 }

```

`\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
5713 \def\part@prefix{}
5714 \@ifpackageloaded{document-structure}{}{
5715   \str_case:VnF \__notesslidesstopsect {
5716     {part}{
5717       \int_set:Nn \l_document_structure_section_level_int {0}
5718       \def\thesection{\arabic{chapter}.\arabic{section}}
5719       \def\part@prefix{\arabic{chapter}.}
5720     }
5721     {chapter}{
5722       \int_set:Nn \l_document_structure_section_level_int {1}
5723       \def\thesection{\arabic{chapter}.\arabic{section}}
5724       \def\part@prefix{\arabic{chapter}.}
5725     }
5726   }{
5727     \int_set:Nn \l_document_structure_section_level_int {2}
5728     \def\part@prefix{}

```

```

5729 }
5730 }
5731
5732 \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

```

5733 \renewenvironment{omgroup}[2][]{
5734   \__document_structure_omgroup_args:n { #1 }
5735   \int_incr:N \l_document_structure_section_level_int
5736   \bool_if:NT \c__notesslides_sectocframes_bool {
5737     \stepcounter{slide}
5738     \begin{frame}[noframenumbering]
5739       \vfill\Large\centering
5740       \red{
5741         \ifcase\l_document_structure_section_level_int\or
5742           \stepcounter{part}
5743           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5744           \def\currentsectionlevel{\omdoc@part@kw}
5745         \or
5746           \stepcounter{chapter}
5747           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5748           \def\currentsectionlevel{\omdoc@chapter@kw}
5749         \or
5750           \stepcounter{section}
5751           \def\__notesslideslabel{\part@prefix\arabic{section}}
5752           \def\currentsectionlevel{\omdoc@section@kw}
5753         \or
5754           \stepcounter{subsection}
5755           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5756           \def\currentsectionlevel{\omdoc@subsection@kw}
5757         \or
5758           \stepcounter{subsubsection}
5759           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5760           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5761         \or
5762           \stepcounter{paragraph}
5763           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5764           \def\currentsectionlevel{\omdoc@paragraph@kw}
5765         \else
5766           \def\__notesslideslabel{}
5767           \def\currentsectionlevel{\omdoc@paragraph@kw}
5768         \fi% end ifcase
5769         \__notesslideslabel%\sref@label@id\__notesslideslabel
5770         \quad #2%
5771       }%
5772     \vfill%
5773   \end{frame}%
5774 }
5775 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
5776 }{}

```


5777 }

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

```
5778 \def\inserttheorembodyfont{\normalfont}
5779 %\bool_if:NF \c__notesslides_notes_bool {
5780 % \defbeamertemplate{theorem begin}{miko}
5781 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5782 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5783 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5784 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
5785 % \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.

```
5786 % \expandafter\def\csname Parent2\endcsname{}
5787 %}
5788
5789 \AddToHook{begindocument}{% this does not work for some reasons
5790 \setbeamertemplate{theorems}[ams style]
5791 }
5792 \bool_if:NT \c__notesslides_notes_bool {
5793 \renewenvironment{columns}[1][{}]{%
5794 \par\noindent%
5795 \begin{minipage}%
5796 \slidewidth\centering\leavevmode%
5797 }{%
5798 \end{minipage}\par\noindent%
5799 }%
5800 \newsavebox\columnbox%
5801 \renewenvironment<>{column}[2][{}]{%
5802 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5803 }{%
5804 \end{minipage}\end{lrbox}\usebox\columnbox%
5805 }%
5806 }
5807 \bool_if:NTF \c__notesslides_noproblems_bool {
5808 \newenvironment{problems}{}{}
5809 }{
5810 \excludecomment{problems}
5811 }
```

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.

```
5812 \gdef\printexcursions{}
5813 \newcommand\excursionref[2]{% label, text
5814 \bool_if:NT \c__notesslides_notes_bool {
5815 \begin{sparagraph}[title=Excursion]
```

```

5816     #2 \sref[fallback=the appendix]{#1}.
5817 \end{sparagraph}
5818 }
5819 }
5820 \newcommand\activate@excursion[2][]{
5821 \gappto\printexcursions{\inputref[#1]{#2}}
5822 }
5823 \newcommand\excursion[4][]{% repos, label, path, text
5824 \bool_if:NT \c__notesslides_notes_bool {
5825 \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5826 }
5827 }

```

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

\excursiongroup

```

5828 \keys_define:nn{notesslides / excursiongroup }{
5829 id .str_set_x:N = \l__notesslides_excursion_id_str,
5830 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5831 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5832 }
5833 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5834 \tl_clear:N \l__notesslides_excursion_intro_tl
5835 \str_clear:N \l__notesslides_excursion_id_str
5836 \str_clear:N \l__notesslides_excursion_mhrepos_str
5837 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5838 }
5839 \newcommand\excursiongroup[1][]{
5840 \__notesslides_excursion_args:n{ #1 }
5841 \ifdefempty\printexcursions{}% only if there are excursions
5842 {\begin{note}
5843 \begin{omgroup}[#1]{Excursions}%
5844 \ifdefempty\l__notesslides_excursion_intro_tl{\{
5845 \inputref[\l__notesslides_excursion_mhrepos_str]{
5846 \l__notesslides_excursion_intro_tl
5847 }
5848 }
5849 \printexcursions%
5850 \end{omgroup}
5851 \end{note}}
5852 }
5853 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5854 \</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.

```
5855 <*package>
5856 <@@=problems>
5857 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5858 \RequirePackage{l3keys2e,expl-keystr-compatible}
5859
5860 \keys_define:nn { problem / pkg }{
5861   notes      .default:n    = { true },
5862   notes      .bool_set:N   = \c__problems_notes_bool,
5863   gnotes     .default:n    = { true },
5864   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5865   hints      .default:n    = { true },
5866   hints      .bool_set:N   = \c__problems_hints_bool,
5867   solutions  .default:n    = { true },
5868   solutions  .bool_set:N   = \c__problems_solutions_bool,
5869   pts        .default:n    = { true },
5870   pts        .bool_set:N   = \c__problems_pts_bool,
5871   min        .default:n    = { true },
5872   min        .bool_set:N   = \c__problems_min_bool,
5873   boxed      .default:n    = { true },
5874   boxed      .bool_set:N   = \c__problems_boxed_bool,
5875   unknown    .code:n       = {}
5876 }
5877 \newif\ifsolutions
5878
5879 \ProcessKeysOptions{ problem / pkg }
5880 \bool_if:NTF \c__problems_solutions_bool {
5881   \solutionstrue
5882 }{
5883   \solutionsfalse
5884 }
```

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

```
5885 \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.

```
5886 \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.

```
5887 \def\prob@problem@kw{Problem}
5888 \def\prob@solution@kw{Solution}
5889 \def\prob@hint@kw{Hint}
5890 \def\prob@note@kw{Note}
5891 \def\prob@gnote@kw{Grading}
5892 \def\prob@pt@kw{pt}
5893 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5894 \AddToHook{begindocument}{
5895   \ltx@ifpackageloaded{babel}{
5896     \makeatletter
5897     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5898     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5899       \input{problem-ngerman.ldf}
5900     }
5901     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5902       \input{problem-finnish.ldf}
5903     }
5904     \clist_if_in:NnT \l_tmpa_clist {french}{
5905       \input{problem-french.ldf}
5906     }
5907     \clist_if_in:NnT \l_tmpa_clist {russian}{
5908       \input{problem-russian.ldf}
5909     }
5910     \makeatother
5911   }{ }
5912 }
```

40.2 Problems and Solutions

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

```
5913 \keys_define:nn{ problem / problem }{
5914   id      .str_set:x:N = \l__problems_prob_id_str,
5915   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5916   min     .tl_set:N    = \l__problems_prob_min_tl,
5917   title   .tl_set:N    = \l__problems_prob_title_tl,
5918   type    .tl_set:N    = \l__problems_prob_type_tl,
5919   refnum  .int_set:N    = \l__problems_prob_refnum_int
5920 }
5921 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5922 \str_clear:N \l__problems_prob_id_str
5923 \tl_clear:N \l__problems_prob_pts_tl
5924 \tl_clear:N \l__problems_prob_min_tl
5925 \tl_clear:N \l__problems_prob_title_tl
5926 \tl_clear:N \l__problems_prob_type_tl
5927 \int_zero_new:N \l__problems_prob_refnum_int
5928 \keys_set:nn { problem / problem }{ #1 }
5929 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5930   \let\l__problems_prob_refnum_int\undefined
5931 }
5932 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5933 \newcounter{problem}
5934 \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.

```

5935 \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

```

5936 \newcommand\prob@number{
5937   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5938     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5939   }{
5940     \int_if_exist:NTF \l__problems_prob_refnum_int {
5941       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5942     }{
5943       \prob@label\theproblem
5944     }
5945   }
5946 }

```

(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.

```

5947 \newcommand\prob@title[3]{%
5948   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5949     #2 \l__problems_inclprob_title_tl #3
5950   }{
5951     \tl_if_exist:NTF \l__problems_prob_title_tl {
5952       #2 \l__problems_prob_title_tl #3
5953     }{
5954       #1
5955     }
5956   }
5957 }

```

(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.

```

5958 \def\prob@heading{
5959   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5960   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5961 }

```

(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`

```

5962 \newenvironment{sproblem}[1][{}]{
5963   \__problems_prob_args:n{#1}%\sref@target%
5964   \@in@omtexttrue% we are in a statement (for inline definitions)
5965   \stepcounter{problem}\record@problem
5966   \def\current@section@level{\prob@problem@kw}
5967   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5968     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5969   }{
5970     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5971   }
5972   \str_if_exist:NTF \l__problems_inclprob_id_str {
5973     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5974   }{
5975     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5976   }
5977
5978
5979   \clist_set:No \l_tmpa_clist \sproblemtype
5980   \tl_clear:N \l_tmpa_tl
5981   \clist_map_inline:Nn \l_tmpa_clist {
5982     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5983       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5984     }
5985   }
5986   \tl_if_empty:NTF \l_tmpa_tl {
5987     \__problems_sproblem_start:
5988   }{
5989     \l_tmpa_tl
5990   }
5991   \stex_ref_new_doc_target:n \sproblemid
5992 }{
5993   \clist_set:No \l_tmpa_clist \sproblemtype
5994   \tl_clear:N \l_tmpa_tl
5995   \clist_map_inline:Nn \l_tmpa_clist {
5996     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5997       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5998     }

```

```

5999 }
6000 \tl_if_empty:NTF \l_tmpa_tl {
6001   \__problems_sproblem_end:
6002 }{
6003   \l_tmpa_tl
6004 }
6005
6006
6007 \smallskip
6008 }
6009
6010
6011 \cs_new_protected:Nn \__problems_sproblem_start: {
6012   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6013 }
6014 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6015
6016 \newcommand\stexpatchproblem[3]{} {
6017   \str_set:Nx \l_tmpa_str{ #1 }
6018   \str_if_empty:NTF \l_tmpa_str {
6019     \tl_set:Nn \__problems_sproblem_start: { #2 }
6020     \tl_set:Nn \__problems_sproblem_end: { #3 }
6021   }{
6022     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6023     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6024   }
6025 }
6026
6027
6028 \bool_if:NT \c__problems_boxed_bool {
6029   \surroundwithmdframed{problem}
6030 }

```

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

```

6031 \def\record@problem{
6032   \protected@write\@auxout{}
6033   {
6034     \string\@problem{\prob@number}
6035     {
6036       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6037         \l__problems_inclprob_pts_tl
6038       }{
6039         \l__problems_prob_pts_tl
6040       }
6041     }%
6042     {
6043       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6044         \l__problems_inclprob_min_tl
6045       }{
6046         \l__problems_prob_min_tl
6047       }
6048     }
6049   }
6050 }

```

(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).

```
6051 \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.

```
6052 \keys_define:nn { problem / solution }{
6053   id               .str_set_x:N = \l__problems_solution_id_str ,
6054   for              .tl_set:N    = \l__problems_solution_for_tl ,
6055   height           .dim_set:N   = \l__problems_solution_height_dim ,
6056   creators         .clist_set:N = \l__problems_solution_creators_clist ,
6057   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
6058   src cite        .tl_set:N     = \l__problems_solution_src cite_tl
6059 }
6060 \cs_new_protected:Nn \__problems_solution_args:n {
6061   \str_clear:N \l__problems_solution_id_str
6062   \tl_clear:N \l__problems_solution_for_tl
6063   \tl_clear:N \l__problems_solution_src cite_tl
6064   \clist_clear:N \l__problems_solution_creators_clist
6065   \clist_clear:N \l__problems_solution_contributors_clist
6066   \dim_zero:N \l__problems_solution_height_dim
6067   \keys_set:nn { problem / solution }{ #1 }
6068 }
```

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

```
6069 \newcommand\@startsolution[1][ ]{
6070   \__problems_solution_args:n { #1 }
6071   \@in@omtexttrue% we are in a statement.
6072   \bool_if:NF \c__problems_boxed_bool { \hrule }
6073   \smallskip\noindent
6074   {\textbf\prob@solution@kw : \enspace}
6075   \begin{small}
6076   \def\current@section@level{\prob@solution@kw}
6077   \ignorespacesandpars
6078 }
```

\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.

```
6079 \newcommand\startsolutions{
6080   \specialcomment{solution}{\@startsolution}{
6081     \bool_if:NF \c__problems_boxed_bool {
6082       \hrule\medskip
6083     }
6084     \end{small}%
6085   }
6086   \bool_if:NT \c__problems_boxed_bool {
6087     \surroundwithmdframed{solution}
6088   }
6089 }
```


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

\stopsolutions

```
6090 \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.

```
6091 \ifsolutions
6092   \startsolutions
6093 \else
6094   \stopsolutions
6095 \fi
```

exnote

```
6096 \bool_if:NTF \c__problems_notes_bool {
6097   \newenvironment{exnote}[1][]{
6098     \par\smallskip\hrule\smallskip
6099     \noindent\textbf{\prob@note@kw : }\small
6100   }{
6101     \smallskip\hrule
6102   }
6103 }{
6104   \excludecomment{exnote}
6105 }
```

hint

```
6106 \bool_if:NTF \c__problems_notes_bool {
6107   \newenvironment{hint}[1][]{
6108     \par\smallskip\hrule\smallskip
6109     \noindent\textbf{\prob@hint@kw :~ }\small
6110   }{
6111     \smallskip\hrule
6112   }
6113 \newenvironment{exhint}[1][]{
6114   \par\smallskip\hrule\smallskip
6115   \noindent\textbf{\prob@hint@kw :~ }\small
6116 }{
6117   \smallskip\hrule
6118 }
6119 }{
6120   \excludecomment{hint}
6121   \excludecomment{exhint}
6122 }
```

gnote

```
6123 \bool_if:NTF \c__problems_notes_bool {
6124   \newenvironment{gnote}[1][]{
6125     \par\smallskip\hrule\smallskip
6126     \noindent\textbf{\prob@gnote@kw : }\small
6127   }{
6128     \smallskip\hrule
6129   }
6130 }{
6131   \excludecomment{gnote}
6132 }
```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6133 \newenvironment{mcb}{
6134   \begin{enumerate}
6135 }{
6136   \end{enumerate}
6137 }
```

we define the keys for the mcc macro

```
6138 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6139   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6140     \bool_set_true:N #1
6141   }{
6142     \bool_set_false:N #1
6143   }
6144 }
6145 \keys_define:nn { problem / mcc }{
6146   id          .str_set:x:N = \l__problems_mcc_id_str ,
6147   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6148   T           .default:n   = { true } ,
6149   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6150   F           .default:n   = { true } ,
6151   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6152   Ttext       .code:n      = {
6153     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6154   } ,
6155   Ftext       .code:n      = {
6156     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6157   }
6158 }
6159 \cs_new_protected:Nn \l__problems_mcc_args:n {
6160   \str_clear:N \l__problems_mcc_id_str
6161   \tl_clear:N \l__problems_mcc_feedback_tl
6162   \bool_set_true:N \l__problems_mcc_t_bool
6163   \bool_set_true:N \l__problems_mcc_f_bool
6164   \bool_set_true:N \l__problems_mcc_Ttext_bool
6165   \bool_set_false:N \l__problems_mcc_Ftext_bool
6166   \keys_set:nn { problem / mcc }{ #1 }
6167 }
```

\mcc

```
6168 \newcommand\mcc[2][] {
6169   \l__problems_mcc_args:n{ #1 }
6170   \item #2
6171   \ifsolutions
6172     \\\
6173     \bool_if:NT \l__problems_mcc_t_bool {
6174       % TODO!
6175       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6176     }
6177     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

6178      % TODO!
6179      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6180    }
6181    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6182      !
6183    }{
6184      \l__problems_mcc_feedback_tl
6185    }
6186    \fi
6187  } %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.

```

6188
6189 \keys_define:nn{ problem / inclproblem }{
6190   id      .str_set:N = \l__problems_inclprob_id_str,
6191   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6192   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6193   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6194   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6195   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6196   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
6197 }
6198 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6199   \str_clear:N \l__problems_prob_id_str
6200   \tl_clear:N \l__problems_inclprob_pts_tl
6201   \tl_clear:N \l__problems_inclprob_min_tl
6202   \tl_clear:N \l__problems_inclprob_title_tl
6203   \tl_clear:N \l__problems_inclprob_type_tl
6204   \int_zero_new:N \l__problems_inclprob_refnum_int
6205   \str_clear:N \l__problems_inclprob_mhrepos_str
6206   \keys_set:nn { problem / inclproblem }{ #1 }
6207   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6208     \let\l__problems_inclprob_pts_tl\undefined
6209   }
6210   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6211     \let\l__problems_inclprob_min_tl\undefined
6212   }
6213   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6214     \let\l__problems_inclprob_title_tl\undefined
6215   }
6216   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6217     \let\l__problems_inclprob_type_tl\undefined
6218   }
6219   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6220     \let\l__problems_inclprob_refnum_int\undefined
6221   }
6222 }

```

```

6223
6224 \cs_new_protected:Nn \__problems_inclprob_clear: {
6225   \let\l__problems_inclprob_id_str\undefined
6226   \let\l__problems_inclprob_pts_tl\undefined
6227   \let\l__problems_inclprob_min_tl\undefined
6228   \let\l__problems_inclprob_title_tl\undefined
6229   \let\l__problems_inclprob_type_tl\undefined
6230   \let\l__problems_inclprob_refnum_int\undefined
6231   \let\l__problems_inclprob_mhrepos_str\undefined
6232 }
6233 \__problems_inclprob_clear:
6234
6235 \newcommand\includeproblem[2][ ]{
6236   \__problems_inclprob_args:n{ #1 }
6237   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6238     \input{#2}
6239   }{
6240     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6241       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6242     }
6243   }
6244   \__problems_inclprob_clear:
6245 }

```

(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.

```

6246 \AddToHook{enddocument}{
6247   \bool_if:NT \c__problems_pts_bool {
6248     \message{Total:~\arabic{pts}~points}
6249   }
6250   \bool_if:NT \c__problems_min_bool {
6251     \message{Total:~\arabic{min}~minutes}
6252   }
6253 }

```

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

```

6254 \def\pts#1{
6255   \bool_if:NT \c__problems_pts_bool {
6256     \marginpar{#1~\prob@pt@kw}
6257   }
6258 }
6259 \def\min#1{
6260   \bool_if:NT \c__problems_min_bool {
6261     \marginpar{#1~\prob@min@kw}
6262   }
6263 }

```

`\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.

```

6264 \newcounter{pts}
6265 \def\show@pts{
6266   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6267     \bool_if:NT \c__problems_pts_bool {
6268       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6269       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6270     }
6271   }{
6272     \tl_if_exist:NT \l__problems_prob_pts_tl {
6273       \bool_if:NT \c__problems_pts_bool {
6274         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6275         \addtocounter{pts}{\l__problems_prob_pts_tl}
6276       }
6277     }
6278   }
6279 }

```

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

`\show@min`

```

6280 \newcounter{min}
6281 \def\show@min{
6282   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6283     \bool_if:NT \c__problems_min_bool {
6284       \marginpar{\l__problems_inclprob_min_tl\ min}
6285       \addtocounter{min}{\l__problems_inclprob_min_tl}
6286     }
6287   }{
6288     \tl_if_exist:NT \l__problems_prob_min_tl {
6289       \bool_if:NT \c__problems_min_bool {
6290         \marginpar{\l__problems_prob_min_tl\ min}
6291         \addtocounter{min}{\l__problems_prob_min_tl}
6292       }
6293     }
6294   }
6295 }
6296 </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.

```
6297 <@@=hwexam>
6298 <*cls>
6299 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6300 \RequirePackage{l3keys2e,expl-keystr-compatible}
6301 \DeclareOption*{
6302   \PassOptionsToClass{\CurrentOption}{document-structure}
6303   \PassOptionsToPackage{\CurrentOption}{stex}
6304   \PassOptionsToPackage{\CurrentOption}{hwexam}
6305   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6306 }
6307 \ProcessOptions
```

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

```
6308 \LoadClass{document-structure}
6309 \RequirePackage{stex}
6310 \RequirePackage{hwexam}
6311 \RequirePackage{tikzinput}
6312 \RequirePackage{graphicx}
6313 \RequirePackage{a4wide}
6314 \RequirePackage{amssymb}
6315 \RequirePackage{amstext}
6316 \RequirePackage{amsmath}
```

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

```

6317 \newcommand\assig@default@type{\hwexam@assignment@kw}
6318 \def\document@hwexamtype{\assig@default@type}
6319 <@@=document_structure>
6320 \keys_define:nn { document-structure / document }{
6321 id .str_set_x:N = \c_document_structure_document_id_str,
6322 hwexamtype .tl_set:N = \document@hwexamtype
6323 }
6324 <@@=hwexam>
6325 </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.

```
6326 \*package>
6327 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6328 \RequirePackage{l3keys2e,expl-keystr-compat}
6329
6330 \newif\iftest\testfalse
6331 \DeclareOption{test}{\testtrue}
6332 \newif\ifmultiple\multiplefalse
6333 \DeclareOption{multiple}{\multipletrue}
6334 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6335 \ProcessOptions
```

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

```
6336 \RequirePackage{keyval}[1997/11/10]
6337 \RequirePackage{problem}
```

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

```
6338 \newcommand\hwexam@assignment@kw{Assignment}
6339 \newcommand\hwexam@given@kw{Given}
6340 \newcommand\hwexam@due@kw{Due}
6341 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6342 blank~for~extra~space}
6343 \def\hwexam@minutes@kw{minutes}
6344 \newcommand\correction@probs@kw{prob.}
6345 \newcommand\correction@pts@kw{total}
6346 \newcommand\correction@reached@kw{reached}
6347 \newcommand\correction@sum@kw{Sum}
6348 \newcommand\correction@grade@kw{grade}
6349 \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

```

6350 \AddToHook{begindocument}{
6351 \ltx@ifpackageloaded{babel}{
6352 \makeatletter
6353 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6354 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6355 \input{hwexam-ngerman.ldf}
6356 }
6357 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6358 \input{hwexam-finnish.ldf}
6359 }
6360 \clist_if_in:NnT \l_tmpa_clist {french}{
6361 \input{hwexam-french.ldf}
6362 }
6363 \clist_if_in:NnT \l_tmpa_clist {russian}{
6364 \input{hwexam-russian.ldf}
6365 }
6366 \makeatother
6367 }{}
6368 }
6369

```

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.

```

6370 \newcounter{assignment}
6371 \numberproblemsin{assignment}
6372 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6373 \keys_define:nn { hwexam / assignment } {
6374 id .str_set:N = \l__hwexam_assign_id_str,
6375 number .int_set:N = \l__hwexam_assign_number_int,
6376 title .tl_set:N = \l__hwexam_assign_title_tl,
6377 type .tl_set:N = \l__hwexam_assign_type_tl,
6378 given .tl_set:N = \l__hwexam_assign_given_tl,
6379 due .tl_set:N = \l__hwexam_assign_due_tl,
6380 loadmodules .code:n = {
6381 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6382 }
6383 }
6384 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6385 \str_clear:N \l__hwexam_assign_id_str
6386 \int_set:Nn \l__hwexam_assign_number_int {-1}
6387 \tl_clear:N \l__hwexam_assign_title_tl
6388 \tl_clear:N \l__hwexam_assign_type_tl
6389 \tl_clear:N \l__hwexam_assign_given_tl
6390 \tl_clear:N \l__hwexam_assign_due_tl
6391 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6392 \keys_set:nn { hwexam / assignment }{ #1 }
6393 }

```

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.

```

6394 \newcommand\given@due[2]{
6395 \bool_lazy_all:nF {
6396 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6397 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6398 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6399 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6400 }{ #1 }
6401
6402 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6403 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6404 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6405 }
6406 }{
6407 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6408 }
6409
6410 \bool_lazy_or:nnF {
6411 \bool_lazy_and_p:nn {
6412 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6413 }{
6414 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6415 }
6416 }{
6417 \bool_lazy_and_p:nn {
6418 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6419 }{
6420 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6421 }
6422 }{ ,~ }
6423
6424 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6425 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6426 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6427 }
6428 }{
6429 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6430 }
6431
6432 \bool_lazy_all:nF {
6433 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6434 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6435 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6436 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6437 }{ #2 }
6438 }

```

`\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.

```

6439 \newcommand\assignment@title[3]{
6440 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
6441 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6442 #1
6443 }{
6444 #2\l__hwexam_assign_title_tl#3
6445 }
6446 }{
6447 #2\l__hwexam_inclassassign_title_tl#3
6448 }
6449 }

```

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

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

```

6450 \newcommand\assignment@number{
6451 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
6452 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6453 \arabic{assignment}
6454 } {
6455 \int_use:N \l__hwexam_assign_number_int
6456 }
6457 }{
6458 \int_use:N \l__hwexam_inclassassign_number_int
6459 }
6460 }

```

(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.

```

6461 \newenvironment{assignment}[1][ ]{
6462 \__hwexam_assignment_args:n { #1 }
6463 %\sref@target
6464 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6465 \global\stepcounter{assignment}
6466 }{
6467 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6468 }
6469 \setcounter{problem}{0}
6470 \def\current@section@level{\document@hwexamtype}
6471 %\sref@label@id{\document@hwexamtype \thesection}
6472 \begin{@assignment}
6473 }{
6474 \end{@assignment}
6475 }

```

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

```

6476 \def\ass@title{
6477 \protect\document@hwexamtype~\arabic{assignment}
6478 \assignment@title{}\{;\}()\;\} -- \given@due{}\{
6479 }
6480 \ifmultiple
6481 \newenvironment{@assignment}{
6482 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6483 \begin{omgroup}[loadmodules]{\ass@title}
6484 }{
6485 \begin{omgroup}{\ass@title}
6486 }
6487 }{
6488 \end{omgroup}
6489 }

```

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

```

6490 \else
6491 \newenvironment{@assignment}{
6492 \begin{center}\bf
6493 \Large@title\strut\
6494 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}()\;\}
6495 \large\given@due{--;\}{}\{;\}
6496 \end{center}
6497 }{}
6498 \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.

```

6499 \keys_define:nn { hwexam / inclassignment } {
6500 %id .str_set_x:N = \l__hwexam_assign_id_str,
6501 number .int_set:N = \l__hwexam_inclassign_number_int,
6502 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6503 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6504 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6505 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6506 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6507 }
6508 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6509 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6510 \tl_clear:N \l__hwexam_inclassign_title_tl
6511 \tl_clear:N \l__hwexam_inclassign_type_tl
6512 \tl_clear:N \l__hwexam_inclassign_given_tl
6513 \tl_clear:N \l__hwexam_inclassign_due_tl
6514 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6515 \keys_set:nn { hwexam / inclassignment }{ #1 }
6516 }
6517 \__hwexam_inclassignment_args:n {}
6518
6519 \newcommand\inputassignment[2][{}]{

```

```

6520 \_hwexam_inclassnment_args:n { #1 }
6521 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6522 \input{#2}
6523 }{
6524 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6525 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6526 }
6527 }
6528 \_hwexam_inclassnment_args:n {}
6529 }
6530 \newcommand\includeassignment[2][]{
6531 \newpage
6532 \inputassignment[#1]{#2}
6533 }

```

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

42.4 Typesetting Exams

\quizheading

```

6534 \ExplSyntaxOff
6535 \newcommand\quizheading[1]{%
6536 \def\@tas{#1}%
6537 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6538 \ifx\@tas\@empty\else%
6539 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6540 \fi%
6541 }
6542 \ExplSyntaxOn

```

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

\testheading

```

6543
6544 \def\hwexamheader{\input{hwexam-default.header}}
6545
6546 \def\hwexamminutes{
6547 \tl_if_empty:NTF \testheading@duration {
6548 {\testheading@min}~\hwexam@minutes@kw
6549 }{
6550 \testheading@duration
6551 }
6552 }
6553
6554 \keys_define:nn { hwexam / testheading } {
6555 min .tl_set:N = \testheading@min,
6556 duration .tl_set:N = \testheading@duration,
6557 reqpts .tl_set:N = \testheading@reqpts,
6558 tools .tl_set:N = \testheading@tools
6559 }
6560 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6561 \tl_clear:N \testheading@min
6562 \tl_clear:N \testheading@duration

```

```

6563 \tl_clear:N \testheading@reqpts
6564 \tl_clear:N \testheading@tools
6565 \keys_set:nn { hwexam / testheading }{ #1 }
6566 }
6567 \newenvironment{testheading}[1][]{
6568   \_hwexam_testheading_args:n{ #1 }
6569   \newcount\check@time\check@time=\testheading@min
6570   \advance\check@time by -\theassignment@totalmin
6571   \newif\if@bonuspoints
6572   \tl_if_empty:NTF \testheading@reqpts {
6573     \@bonuspointsfalse
6574   }{
6575     \newcount\bonus@pts
6576     \bonus@pts=\theassignment@totalpts
6577     \advance\bonus@pts by -\testheading@reqpts
6578     \edef\bonus@pts{\the\bonus@pts}
6579     \@bonuspointstrue
6580   }
6581   \edef\check@time{\the\check@time}
6582
6583   \makeatletter\hwexamheader\makeatother
6584 }{
6585   \newpage
6586 }

```

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

\testspace

```

6587 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6588 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6589 \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.

```

6590 <@=problems>
6591 \renewcommand\@problem[3]{
6592   \stepcounter{assignment@probs}
6593   \def\__problemspts{#2}
6594   \ifx\__problemspts\@empty\else
6595     \addtocounter{assignment@totalpts}{#2}
6596   \fi
6597   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6598   \xdef\correction@probs{\correction@probs & #1}%
6599   \xdef\correction@pts{\correction@pts & #2}
6600   \xdef\correction@reached{\correction@reached &}

```

```

6601 }
6602 <@@=hwexam>

```

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

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

```

6603 \newcounter{assignment@probs}
6604 \newcounter{assignment@totalpts}
6605 \newcounter{assignment@totalmin}
6606 \def\correction@probs{\correction@probs@kw}
6607 \def\correction@pts{\correction@pts@kw}
6608 \def\correction@reached{\correction@reached@kw}
6609 \stepcounter{assignment@probs}
6610 \newcommand\correction@table{
6611 \resizebox{\textwidth}{!}{%
6612 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6613 &\multicolumn{\theassignment@probs}{c|}|%|
6614 {\footnotesize\correction@forgrading@kw} &\\ \hline
6615 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6616 \correction@pts & \theassignment@totalpts & \\ \hline
6617 \correction@reached & & \[.7cm]\hline
6618 \end{tabular}}
6619 </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}

```