

The sTeX3 Package *

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

2022-01-31

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 2022-01-31)

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	5
4	Creating New Modules and Symbols	6
4.1	sTeX Archives	6
4.1.1	The Local MathHub-Directory	6
4.1.2	The Structure of sTeX Archives	6
4.2	Primitive Symbols (The sTeX Metatheory)	7
5	sTeX Statements (Definitions, Theorems, Examples, ...)	8
6	Additional Packages	9
6.1	Modular Document Structuring	9
6.2	Slides and Course Notes	9
6.3	Homework, Problems and Exams	9
7	Stuff	10
7.1	Modules	10
7.1.1	Semantic Macros and Notations	10
	Other Argument Types	12
	Precedences	14
7.1.2	Archives and Imports	14
	Namespaces	14
	Paths in Import-Statements	15
II	Documentation	16
8	sTeX-Basics	17
8.1	Macros and Environments	17
9	sTeX-MathHub	19
9.1	Macros and Environments	19
9.1.1	Files, Paths, URIs	19
9.1.2	MathHub Archives	20
10	sTeX-References	22
10.1	Macros and Environments	22

11	sTeX-Modules	23
11.1	Macros and Environments	23
11.1.1	The <code>module</code> -environment	25
12	sTeX-Module Inheritance	28
12.1	Macros and Environments	28
12.1.1	SMS Mode	28
12.1.2	Imports and Inheritance	29
13	sTeX-Symbols	32
13.1	Macros and Environments	32
14	sTeX-Terms	35
14.1	Macros and Environments	35
15	sTeX-Structural Features	38
15.1	Macros and Environments	38
15.1.1	Structures	38
16	sTeX-Statements	39
16.1	Macros and Environments	39
17	sTeX-Proofs: Structural Markup for Proofs	40
17.1	Introduction	42
17.2	The User Interface	43
17.2.1	Package Options	43
17.2.2	Proofs and Proof steps	43
17.2.3	Justifications	43
17.2.4	Proof Structure	44
17.2.5	Proof End Markers	45
17.2.6	Configuration of the Presentation	45
17.3	Limitations	45
18	sTeX-Metatheory	47
18.1	Symbols	47
III	Extensions	48
19	Tikzinput	49
19.1	Macros and Environments	49

20	document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX	50
20.1	Introduction	50
20.2	The User Interface	51
20.2.1	Package and Class Options	51
20.2.2	Document Structure	51
20.2.3	Ignoring Inputs	52
20.2.4	Structure Sharing	53
20.2.5	Global Variables	53
20.2.6	Colors	54
20.3	Limitations	54
21	Slides and Course Notes	55
21.1	Introduction	55
21.2	The User Interface	55
21.2.1	Package Options	55
21.2.2	Notes and Slides	56
21.2.3	Header and Footer Lines of the Slides	57
21.2.4	Frame Images	57
21.2.5	Colors and Highlighting	58
21.2.6	Front Matter, Titles, etc.	58
21.2.7	Excursions	58
21.2.8	Miscellaneous	58
21.3	Limitations	58
22	problem.sty: An Infrastructure for formatting Problems	59
22.1	Introduction	59
22.2	The User Interface	59
22.2.1	Package Options	59
22.2.2	Problems and Solutions	60
22.2.3	Multiple Choice Blocks	61
22.2.4	Including Problems	61
22.2.5	Reporting Metadata	61
22.3	Limitations	61
23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	63
23.1	Introduction	64
23.2	The User Interface	64
23.2.1	Package and Class Options	64
23.2.2	Assignments	64
23.2.3	Typesetting Exams	64
23.2.4	Including Assignments	65
23.3	Limitations	65
IV	Implementation	67

24	<code>STeX</code>-Basics Implementation	68
24.1	The <code>STeXDocument</code> Class	68
24.2	Preliminaries	68
24.3	Messages and logging	69
24.4	Persistence	70
24.5	HTML Annotations	70
24.6	Languages	73
24.7	Activating/Deactivating Macros	74
25	<code>STeX</code>-MathHub Implementation	76
25.1	Generic Path Handling	76
25.2	PWD and <code>kpsewhich</code>	78
25.3	File Hooks and Tracking	79
25.4	MathHub Repositories	80
26	<code>STeX</code>-References Implementation	86
26.1	Document URIs and URLs	86
26.2	Setting Reference Targets	88
26.3	Using References	89
27	<code>STeX</code>-Modules Implementation	91
27.1	The module environment	94
27.2	Invoking modules	100
28	<code>STeX</code>-Module Inheritance Implementation	102
28.1	SMS Mode	102
28.2	Inheritance	106
29	<code>STeX</code>-Symbols Implementation	111
29.1	Symbol Declarations	111
29.2	Notations	117
30	<code>STeX</code>-Terms Implementation	126
30.1	Symbol Invocations	126
30.2	Terms	129
30.3	Notation Components	135
31	<code>STeX</code>-Structural Features Implementation	138
31.1	Imports with modification	138
31.2	The feature environment	139
31.3	Features	140
32	<code>STeX</code>-Statements Implementation	146
32.1	Definitions	146
32.2	Assertions	149
32.3	Examples	151
32.4	Logical Paragraphs	153

33 The Implementation	156
33.1 Package Options	156
33.2 Proofs	156
33.3 Justifications	162
34 \LaTeX-Others Implementation	164
35 \LaTeX-Metatheory Implementation	165
36 Tikzinput Implementation	168
37 document-structure.sty Implementation	170
37.1 The OMDoc Class	170
37.2 Class Options	170
37.3 Beefing up the <code>document</code> environment	171
37.4 Implementation: OMDoc Package	171
37.5 Package Options	171
37.6 Document Structure	173
37.7 Front and Backmatter	176
37.8 Global Variables	178
38 MiKoSlides – Implementation	179
38.1 Class and Package Options	179
38.2 Notes and Slides	181
38.3 Header and Footer Lines	185
38.4 Frame Images	186
38.5 Colors and Highlighting	187
38.6 Sectioning	188
38.7 Excursions	190
39 The Implementation	192
39.1 Package Options	192
39.2 Problems and Solutions	193
39.3 Multiple Choice Blocks	198
39.4 Including Problems	199
39.5 Reporting Metadata	200
40 Implementation: The hwexam Class	202
40.1 Class Options	202
41 Implementation: The hwexam Package	204
41.1 Package Options	204
41.2 Assignments	205
41.3 Including Assignments	208
41.4 Typesetting Exams	209
41.5 Leftovers	211

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.
- **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

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

2.2 A First sTeX Document

Having set everything up, we can write a first sTeX 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}

\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}{
      \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 sTeX *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).

TODO explain `\usemodule`, `\symref`, semantic macros, notation options (`[frac]`).
TODO explain `xhtml` conversion, MMT compilation (requires an archive...?).

³EdNOTE: somewhere later

Chapter 3

Using Semantic Macros

TODO

Chapter 4

Creating New Modules and Symbols

TODO

4.1 $\text{\texttt{S}\TeX}$ Archives

4.1.1 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, $\text{\texttt{S}\TeX}$ uses *archives* that determine the global namespaces for symbols and statements and make it possible for $\text{\texttt{S}\TeX}$ to find content referenced via such URIs.

All $\text{\texttt{S}\TeX}$ archives need to exist in the local MathHub-directory. $\text{\texttt{S}\TeX}$ knows where this folder is via one of three means:

1. If the $\text{\texttt{S}\TeX}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\text{\texttt{S}\TeX}$ 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 $\text{\texttt{S}\TeX}$ -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, $\text{\texttt{S}\TeX}$ 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.

4.1.2 The Structure of $\text{\texttt{S}\TeX}$ Archives

An $\text{\texttt{S}\TeX}$ 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 $\text{\texttt{S}\TeX}$ system, it needs to be in `/user/foo/MathHub/smglom/calculus`.

4.2 Primitive Symbols (The $\text{\texttt{sTeX}}$ Metatheory)

Chapter 5

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

Chapter 6

Additional Packages

6.1 Modular Document Structuring

6.2 Slides and Course Notes

6.3 Homework, Problems and Exams

Chapter 7

Stuff

7.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

7.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

ab

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

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


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

Example 2

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

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

.

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

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

Example 3

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

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

.

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

Example 4

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

Multiplying again by b yields...

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

Example 5

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

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

⁴EdNOTE: TODO

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

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

Example 6

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

The operator $\textcolor{teal}{+}$ adds two elements, as in $a\textcolor{teal}{+}b$.

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

Example 7

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

$\textcolor{teal}{\textit{Multiplication}}$ (denoted by \cdot) is defined by...

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

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

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

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

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

Other Argument Types

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

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

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

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

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

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

Example 8

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

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

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

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

Example 9

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

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

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

5 6

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

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

Precedences

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

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

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

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

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

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

For example:

Example 10

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

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

7.1.2 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 8

sTeX-Basics

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

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

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

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

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

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

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

8.1 Macros and Environments

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

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

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

<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 9

STEX-MathHub

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

9.1 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.
--------------------------------	---

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

9.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 10

sTeX-References

Code related to links and cross-references

10.1 Macros and Environments

Chapter 11

sTeX-Modules

Code related to Modules

11.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

11.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

Test 5

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

```
Module 11.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

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

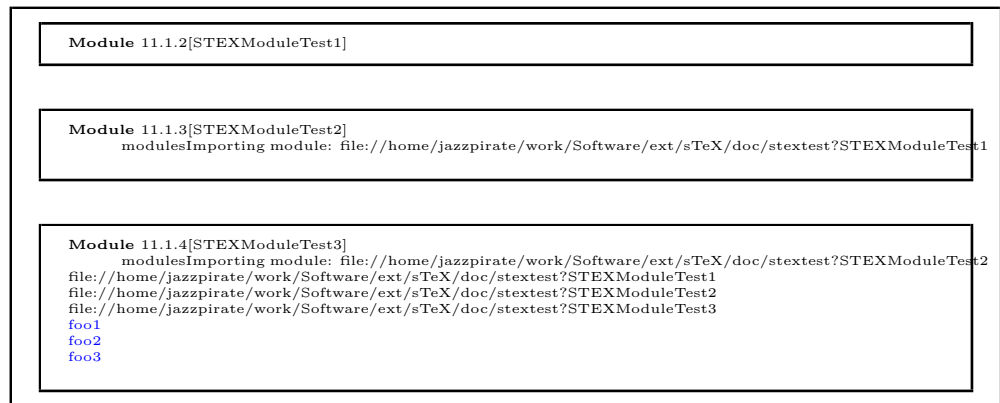
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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

`\stex_activate_module:n`

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

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

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

Test 7

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

12.1.2 Imports and Inheritance

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

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

Test 8

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

Module 12.1.1[Foo]

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

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

Module 12.1.2[Importtest]

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

Module 12.1.3[Importtest2]

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

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

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

Test 9

```

\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:-\present\foo\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:-\present\foo\
Meaning:-\present\bar\
All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,~} \
All symbols:-
\seq_use:Nn \l_stex_all_symbols_seq {,~}
\ExplSyntaxOff
\end{module}

```

Module 12.1.4[UseTest1]

Module 12.1.5[UseTest2]
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextestUseTest1 Meaning: >undefined<

Module 12.1.6[UseTest3]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2 Meaning: >undefined<
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<
All modules: http://mathhub.info/sTeX?Metatheory, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3,
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?seqtype, http://mathhub.info/sTeX?Metatheory?sequence-index, http://mathhub.info/sTeX?Metatheory?aseqfromto, http://mathhub.info/sTeX?Metatheory?aseqfromtovia, http://mathhub.info/sTeX?Metatheory?module-type, http://mathhub.info/sTeX?Metatheory?mathematical-structure, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar

Test 10

```

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

```

Circular dependencies:

Module 12.1.7[CircDep1]
>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

`\stex_import_module_uri:nn`

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

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

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

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

Chapter 13

TeX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

<u><code>\symdecl</code></u>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</code>
------------------------------	--

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 13.1.2[NotationTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo

`\symdef`

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

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

Test 13

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

Module 13.1.3[SymdefTest]
 $a+b+c$

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Test 14

```

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

```

Module 14.1.1[MathTest1]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle a^b_c \rangle$
and $\langle a^b_c \rangle$.

Test 15

```

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

\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}
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\bar{displaystyle} \bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{displaystyle} \bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
\withbrackets[ {  $\bar{displaystyle}$  }
\mult{a,\plus{\frac{ab}{c}}}{ }
\end{module}

```

Module 14.1.2[MathTest2]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle a|[b:c,d:e,f]^{g,h} \rangle$
and $\langle a|[b:c]^{g,h} \rangle$ and $\langle a|[b]^{c,d} \rangle$
 $a+(b \cdot c)$ and $a \cdot \frac{a}{b} + \frac{a}{c}$
 $a+(b \cdot c)$ and $a \cdot \frac{a}{b} + \frac{a}{c}$

`\stex_term_custom:nn`

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

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

```
Module 14.1.3[TextTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo
some a and some b and also some c here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a
```

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 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 17

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

17.1 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.

17.2 The User Interface

17.2.1 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).

17.2.2 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.

17.2.3 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

17.2.4 Proof Structure

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

17.2.5 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{}`.

17.2.6 Configuration of the Presentation

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

EdN:8

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

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

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

Figure 2: Configuration Proof Step Label Styles

17.3 Limitations

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

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

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

Chapter 18

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.

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

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

Chapter 20

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

The `omdoc` package is part of the $\S\TeX$ collection, a version of \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX into a document format for mathematical knowledge management (MKM).

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

20.1 Introduction

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

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

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

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

20.2 The User Interface

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

20.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

The `omdoc` package accepts the same except the first two.

20.2.2 Document Structure

document

\documentkeys

id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble². This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the \LaTeX ML transformation.

omgroup

id

creators

contributors

short

loadmodules

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

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

blindomgroup

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

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

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`

`\CurrentSectionLevel`

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

20.2.3 Ignoring Inputs

`ignore`
`showignores`

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

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

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

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

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

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

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

20.2.4 Structure Sharing

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

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

20.2.5 Global Variables

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

`\ifSGvar` With `\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar`{ $\langle vname \rangle$ }{ $\langle val \rangle$ }{ $\langle ctext \rangle$ } tests the content of the global variable $\langle vname \rangle$, only if (after expansion) it is equal to $\langle val \rangle$, the conditional text $\langle ctext \rangle$ is formatted.

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

20.2.6 Colors

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

20.3 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 21

Slides and Course Notes

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

21.1 Introduction

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

21.2 The User Interface

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

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

21.2.1 Package Options

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

- | | |
|---------------------------|--|
| <code>slides</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 21.2.2). |
| <code>notes</code> | |
| <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 21.2.4). 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>.

21.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

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

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

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

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

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

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

21.2.3 Header and Footer Lines of the Slides

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

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

21.2.4 Frame Images

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

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

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

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

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


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

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

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

`\mhframeimage{baz/foobar}`

21.2.5 Colors and Highlighting

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

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

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

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

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

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

```
\excursionref          Sometimes, we want to reference – in an excursion – part of another. We can use
\excursionref{<label>} for that.
```

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

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

21.2.8 Miscellaneous

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 [sTeXGitHub repository](#) [[sTeX](#)].

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

Chapter 22

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.

22.1 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.

22.2 The User Interface

22.2.1 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

22.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

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

22.2.3 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

22.2.4 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.

22.2.5 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.

22.3 Limitations

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

1. none reported yet

```

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

```

Problem0.0 ()

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

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

Problem0.0 ()

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

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

Example 7: A Problem with a multiple choice block

Chapter 23

`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

23.1 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.

23.2 The User Interface

23.2.1 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].

23.2.2 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 — 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).

23.2.3 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` 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.

23.2.4 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.

23.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 24

ST_EX -Basics Implementation

24.1 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>
```

24.2 Preliminaries

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

Package options:

```

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

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

```

(End definition for `\stex` and `\TeX`. These functions are documented on page 17.)

24.3 Messages and logging

```

43 <@@=stex_log>

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

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

```

(End definition for `\stex_debug:nn`. This function is documented on page 17.)

Redirecting messages:

```

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

```

24.4 Persistence

78 `<@@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

(End definition for `\stex_add_to_sms:n`. This function is documented on page 17.)

24.5 HTML Annotations

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

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

```

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

```

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

```

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

```

```

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

```

(End definition for \if@latexml and \latexml_if:TF. These functions are documented on page 17.)

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

```

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

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

`_stex_annotate_checkempty:n`

```

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

```

(End definition for _stex_annotate_checkempty:n.)

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

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

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`.
These functions are documented on page [18](#).)

24.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 18.)

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

```

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

```

24.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```


(End definition for \stex_deactivate_macro:Nn. This function is documented on page 18.)

\stex_reactivate_macro:N

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

(End definition for \stex_reactivate_macro:N. This function is documented on page 18.)

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

Chapter 25

STEX -MathHub Implementation

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

25.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

```
\stex_path_from_string:Nn
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
297 \cs_new_protected:Nn \stex_path_from_string:Nn {
298   \str_set:Nx \l_tmpa_str { #2 }
299   \str_if_empty:NTF \l_tmpa_str {
300     \seq_clear:N #1
301   }{
302     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
303     \sys_if_platform_windows:T{
304       \seq_clear:N \l_tmpa_tl
305       \seq_map_inline:Nn #1 {
306         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
307         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
308       }
309     }
310   }
```

```

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

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 19.)

```

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

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 19.)

```

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

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves `.` and `..` path segments.

```

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

```

```

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

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 19.)

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

```

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

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 19.)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 19.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 19.)

25.3 File Hooks and Tracking

387 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

388 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

390 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

391 `\c_stex_mainfile_str`

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 19.)

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

392 `\seq_gclear_new:N\g_stex_currentfile_seq`

393 `\AddToHook{file/before}{`

394 `\stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}`

395 `\stex_path_if_absolute:N\g_stex_currentfile_seq{`

396 `\exp_args:Nne\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}`

397 `}{`

398 `\stex_path_from_string:Nn\g_stex_currentfile_seq{`

399 `\c_stex_pwd_str/\CurrentFilePath/\CurrentFile`

400 `}`

401 `}`

402 `\seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq`

403 `\exp_args:Nno\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq`

404 `}`

405 `\AddToHook{file/after}{`

406 `\seq_if_empty:N\g__stex_files_stack{`

407 `\seq_gpop:N\g__stex_files_stack\l_tmpa_seq`

408 `}`

409 `\seq_if_empty:N\g__stex_files_stack{`

410 `\seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq`

411 `}{`

412 `\seq_get:NN\g__stex_files_stack\l_tmpa_seq`

413 `\seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq`

414 `}`

415 `}`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 20.)

25.4 MathHub Repositories

```

416 <@@=stex_mathhub>

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

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 20.)

```

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

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
453 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

`_stex_mathhub_find_manifest:N` Attempts to find the MANIFEST.MF in some file path and stores its path in `\l__stex_mathhub_manifest_file_seq`:

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

489 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

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

```

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```



```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 20.)

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

```

567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \exp_args:Ne \l_tmpa_cs{
572       \prop_item:Nn \l_stex_current_repository_prop { id }
573     }
574   }{
575     \stex_require_repository:n \l_tmpa_str
576     \str_set:Nx \l_tmpa_str { #1 }
577     \exp_args:Nne \use:nn {
578       \stex_set_current_repository:n \l_tmpa_str
579       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
580     }{
581       \stex_set_current_repository:n {
582         \prop_item:Nn \l_stex_current_repository_prop { id }
583       }
584     }
585   }
586 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn

587 \newif \ifinputref \inputreffalse
588
589 \cs_new_protected:Nn \stex_mhinput:nn {
590   \stex_in_repository:nn {#1} {
591     \ifinputref
592       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
593     \else
594       \inputreftrue
595       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
596     \inputreffalse
597   \fi
598 }
599 }
600 \NewDocumentCommand \mhinput { 0{} m}{
601   \stex_mhinput:nn{ #1 }{ #2 }
602 }
603
604 \cs_new_protected:Nn \stex_inputref:nn {
605   \stex_in_repository:nn {#1} {
606     \bool_lazy_any:nTF {
607       {\rustex_if_p:} {\latexml_if_p:}
608     } {
609       \str_clear:N \l_tmpa_str
610       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
611         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
612       }
613       \stex_annotate_invisible:nnn{inputref}{
614         \l_tmpa_str / #2
615       }{}
616     }{
617       \begingroup
618         \inputreftrue
619         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620       \endgroup
621     }
622   }
623 }
624
625 \NewDocumentCommand \inputref { 0{} m}{
626   \stex_inputref:nn{ #1 }{ #2 }
627 }
628
629 \cs_new_protected:Nn \stex_mhbibresource:nn {
630   \stex_in_repository:nn {#1} {
631     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
632   }
633 }
634 \newcommand\addmhbibresource[2] []{
635   \stex_mhbibresource:nn{ #1 }{ #2 }
636 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page [21](#).)

\mhpath

```
637 \def \mhpath #1 #2 {
638   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
639     \c_stex_mathhub_str /
640     \prop_item:Nn \l_stex_current_repository_prop { id }
641     / source / #2
642   }{
643     \c_stex_mathhub_str / #1 / source / #2
644   }
645 }
```

(End definition for \mhpath. This function is documented on page 21.)

\libinput

```
646 \cs_new_protected:Npn \libinput #1 {
647   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
648     \msg_error:nnn{stex}{error/notinarchive}\libinput
649   }
650   \bool_set_false:N \l_tmpa_bool
651   \tl_clear:N \l_tmpa_tl
652   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
653   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
654   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
655   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
656     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
657     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
658       / meta-inf / lib / #1.tex}{
659       \bool_set_true:N \l_tmpa_bool
660       \tl_put_right:Nx \l_tmpa_tl {
661         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
662           / meta-inf / lib / #1.tex}
663       }
664     }{}
665   }
666   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
667     / \l_tmpa_str / lib / #1.tex
668   }{
669     \bool_set_true:N \l_tmpa_bool
670     \tl_put_right:Nx \l_tmpa_tl {
671       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
672         / \l_tmpa_str / lib / #1.tex}
673     }
674   }{}
675   \bool_if:NF \l_tmpa_bool {
676     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
677   }
678   \l_tmpa_tl
679 }
```

(End definition for \libinput. This function is documented on page 21.)

```
680 </package>
```

Chapter 26

STEX -References Implementation

```
681 <*package>
682
683 %%%%%%%%%% references.dtx %%%%%%%%%%
684
685 %\RequirePackage{hyperref}
686 %\RequirePackage{cleveref}
687 <@@=stex_refs>
688
689 Warnings and error messages
690
691 \iow_new:N \c__stex_refs_refs_iow
692 \AddToHook{begindocument}{
693   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
694 }
695 \AddToHook{enddocument}{
696   \iow_close:N \c__stex_refs_refs_iow
697 }
698
699 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
700
701 \NewDocumentCommand \STEXreftitle { m } {
702   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
703 }
```

26.1 Document URIs and URLs

```
702 \seq_new:N \g__stex_refs_all_refs_seq
703
704 \str_new:N \l_stex_current_docns_str
705
706 \cs_new_protected:Nn \stex_get_document_uri: {
707   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
708   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
709   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
710   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

711 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
712
713 \str_clear:N \l_tmpa_str
714 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
715   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
716 }
717
718 \str_if_empty:NTF \l_tmpa_str {
719   \str_set:Nx \l_stex_current_docns_str {
720     file:/\stex_path_to_string:N \l_tmpa_seq
721   }
722 }{
723   \bool_set_true:N \l_tmpa_bool
724   \bool_while_do:Nn \l_tmpa_bool {
725     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
726     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
727       {source} { \bool_set_false:N \l_tmpa_bool }
728     }{}{
729       \seq_if_empty:NT \l_tmpa_seq {
730         \bool_set_false:N \l_tmpa_bool
731       }
732     }
733   }
734
735   \seq_if_empty:NTF \l_tmpa_seq {
736     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
737   }{
738     \str_set:Nx \l_stex_current_docns_str {
739       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
740     }
741   }
742 }
743 }
744
745 \str_new:N \l_stex_current_docurl_str
746 \cs_new_protected:Nn \stex_get_document_url: {
747   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
748   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
749   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
750   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
751   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
752
753   \str_clear:N \l_tmpa_str
754   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
755     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
756       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
757     }
758   }
759
760   \str_if_empty:NTF \l_tmpa_str {
761     \str_set:Nx \l_stex_current_docurl_str {
762       file:/\stex_path_to_string:N \l_tmpa_seq
763     }
764   }{
765     \bool_set_true:N \l_tmpa_bool

```

```

765 \bool_while_do:Nn \l_tmpa_bool {
766   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
767   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
768     {source} { \bool_set_false:N \l_tmpa_bool }
769   }{}{
770     \seq_if_empty:NT \l_tmpa_seq {
771       \bool_set_false:N \l_tmpa_bool
772     }
773   }
774 }
775
776 \seq_if_empty:NTF \l_tmpa_seq {
777   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
778 }{
779   \str_set:Nx \l_stex_current_docurl_str {
780     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
781   }
782 }
783 }
784 }

```

26.2 Setting Reference Targets

```

785 \str_const:Nn \c__stex_refs_url_str{URL}
786 \str_const:Nn \c__stex_refs_ref_str{REF}
787 % @currentlabel -> number
788 % @currentlabelname -> title
789 % @currentHref -> name.number <- id of some kind
790 % \theH# -> \arabic{section}
791 % \the# -> number
792 % \hyper@makecurrent{#}
793 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
794   \stex_get_document_uri:
795   \str_set:Nx \l_tmpa_str { #1 }
796   \str_if_empty:NT \l_tmpa_str {
797     \int_zero:N \l_tmpa_int
798     \bool_set_true:N \l_tmpa_bool
799     \bool_while_do:Nn \l_tmpa_bool {
800       \cs_if_exist:cTF {
801         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
802       }{
803         \int_incr:N \l_tmpa_int
804       }{
805         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
806         \bool_set_false:N \l_tmpa_bool
807       }
808     }
809   }
810   \str_set:Nx \l_tmpa_str {
811     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
812   }
813   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
814   \stex_if_smsmode:TF {
815     \stex_get_document_url:

```

```

816 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
817 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
818 }{
819 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
820 \exp_args:Nx\label{sref_\l_tmpa_str}
821 \str_gset:cx {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
822 }
823 }

824 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
825 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
826 }

```

26.3 Using References

```

827 \str_new:N \l__stex_refs_indocument_str
828 \keys_define:nn { stex / sref } {
829 linktext .tl_set:N = \l__stex_refs_linktext_tl ,
830 fallback .tl_set:N = \l__stex_refs_fallback_tl ,
831 pre .tl_set:N = \l__stex_refs_pre_tl ,
832 post .tl_set:N = \l__stex_refs_post_tl ,
833 %indoc .str_set_x:N = \l__stex_refs_repo_str ,
834 }
835
836 \bool_new:N \c__stex_refs_hyperref_bool
837 \bool_set_false:N \c__stex_refs_hyperref_bool
838 \AddToHook{begindocument}{
839 \@ifpackageloaded{hyperref}{
840 \bool_set_true:N \c__stex_refs_hyperref_bool
841 }{}
842 }
843
844
845 \cs_new_protected:Nn \__stex_refs_args:n {
846 \tl_clear:N \l__stex_refs_linktext_tl
847 \tl_clear:N \l__stex_refs_fallback_tl
848 \tl_clear:N \l__stex_refs_pre_tl
849 \tl_clear:N \l__stex_refs_post_tl
850 \str_clear:N \l__stex_refs_repo_str
851 \keys_set:nn { stex / sref } { #1 }
852 }
853
854 \NewDocumentCommand \sref { 0{} m }{
855 \__stex_refs_args:n { #1 }
856 \str_if_empty:NTF \l__stex_refs_indocument_str {
857 \str_set:Nn \l_tmpa_str { #2 }
858 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
859 \tl_set:Nn \l_tmpa_tl {
860 \l__stex_refs_fallback_tl
861 }
862 \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
863 \str_set:Nn \l_tmpb_str { ##1 }
864 \str_if_eq:eeT { \l_tmpa_str } {
865 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
866 } {

```

```

867 \seq_map_break:n {
868   \tl_set:Nn \l_tmpa_tl {
869     % doc uri in \l_tmpb_str
870     \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str_type}}
871     \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
872       % reference
873       \cs_if_exist:cTF{autoref}{
874         \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
875       }{
876         \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
877       }
878     }{
879       % URL
880       \if_bool:N \c__stex_refs_hyperref_bool {
881         \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
882       }{
883         \l__stex_refs_fallback_tl
884       }
885     }
886   }
887 }
888 }
889 }
890 \l_tmpa_tl
891 }{
892   % TODO
893 }
894 }
895
896 </package>

```


Chapter 27

STEX -Modules Implementation

```
897 <*package>
898
899 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
900
901 <@@=stex_modules>
    Warnings and error messages
902 \msg_new:nnn{stex}{error/unknownmodule}{
903   No~module~#1~found
904 }
905 \msg_new:nnn{stex}{error/syntax}{
906   Syntax~error:~#1
907 }
908 \msg_new:nnn{stex}{error/siglanguage}{
909   Module~#1~declares~signature~#2,~but~does~not~
910   declare~its~language
911 }
912
913 \msg_new:nnn{stex}{error/concllictingmodules}{
914   Conflicting~imports~for~module~#1
915 }

\l_stex_current_module_str The current module:
916 \str_new:N \l_stex_current_module_str
    (End definition for \l_stex_current_module_str. This variable is documented on page 23.)

\l_stex_all_modules_seq Stores all available modules
917 \seq_new:N \l_stex_all_modules_seq
    (End definition for \l_stex_all_modules_seq. This variable is documented on page 23.)

\stex_if_in_module_p:
\stex_if_in_module:TF
918 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
919   \str_if_empty:NTF \l_stex_current_module_str
920     \prg_return_false: \prg_return_true:
921 }
```

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

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

```
922 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
923   \prop_if_exist:cTF { c_stex_module_#1_prop }
924   \prg_return_true: \prg_return_false:
925 }
```

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
926 \cs_new_protected:Nn \stex_add_to_current_module:n {
927   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
928 }
929 \cs_new_protected:Npn \STEXexport {
930   \beginngroup
931   \newlinechar=-1\relax
932   \endlinechar=-1\relax
933   %\catcode'\ = 9\relax
934   \expandafter\endgroup\STEXexport:n
935 }
936 \cs_new_protected:Nn \STEXexport:n {
937   \ignorespaces #1
938   \stex_add_to_current_module:n { \ignorespaces #1 }
939   \stex_smsmode_set_codes:
940 }
941 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 24.)

`\stex_add_constant_to_current_module:n`

```
942 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
943   \str_set:Nx \l_tmpa_str { #1 }
944   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
945 }
946
947 \cs_new_protected:Nn \stex_add_field_to_current_module:n {
948   \str_set:Nx \l_tmpa_str { #1 }
949   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
950 }
```

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

`\stex_collect_imports:nn`

```
951 \cs_new_protected:Nn \stex_collect_imports:nn {
952   \seq_clear:N \l_stex_collect_imports_seq
953   \__stex_modules_collect_imports:n {#1}
954 }
955 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
956   \seq_map_inline:cn {c_stex_module_#1_imports} {
957     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
958       \__stex_modules_collect_imports:n { ##1 }
959     }
960 }
```

```

960 }
961 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
962   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
963 }
964 }

```

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

`\stex_add_import_to_current_module:n`

```

965 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
966   \str_set:Nx \l_tmpa_str { #1 }
967   \exp_args:Nno
968   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
969     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
970   }
971 }

```

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

`\stex_modules_compute_namespace:nN`

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

```

972 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
973   \str_set:Nx \l_tmpa_str { #1 }
974   \seq_set_eq:NN \l_tmpa_seq #2
975   % split off file extension
976   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
977   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
978   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
979   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
980
981   \bool_set_true:N \l_tmpa_bool
982   \bool_while_do:Nn \l_tmpa_bool {
983     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
984     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
985       {source} { \bool_set_false:N \l_tmpa_bool }
986     }{}{
987       \seq_if_empty:NT \l_tmpa_seq {
988         \bool_set_false:N \l_tmpa_bool
989       }
990     }
991   }
992
993   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
994   \str_if_empty:NTF \l_stex_modules_subpath_str {
995     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
996   }{
997     \str_set:Nx \l_stex_modules_ns_str {
998       \l_tmpa_str/\l_stex_modules_subpath_str
999     }
1000   }
1001 }

```

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

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1002 \str_new:N \l_stex_modules_ns_str
1003 \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.

```

1004 \cs_new_protected:Nn \stex_modules_current_namespace: {
1005   \str_clear:N \l_stex_modules_subpath_str
1006   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
1007     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1008   }{
1009     % split off file extension
1010     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1011     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1012     \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1013     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1014     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1015     \str_set:Nx \l_stex_modules_ns_str {
1016       file:/\stex_path_to_string:N \l_tmpa_seq
1017     }
1018   }
1019 }

```

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

27.1 The module environment

module arguments:

```

1020 \keys_define:nn { stex / module } {
1021   title      .str_set_x:N = \l_stex_module_title_str ,
1022   ns         .str_set_x:N = \l_stex_module_ns_str ,
1023   lang       .str_set_x:N = \l_stex_module_lang_str ,
1024   sig        .str_set_x:N = \l_stex_module_sig_str ,
1025   creators   .str_set_x:N = \l_stex_module_creators_str ,
1026   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1027   meta       .str_set_x:N = \l_stex_module_meta_str ,
1028   srccite    .str_set_x:N = \l_stex_module_srccite_str
1029 }
1030
1031 \cs_new_protected:Nn \__stex_modules_args:n {
1032   \str_clear:N \l_stex_module_title_str
1033   \str_clear:N \l_stex_module_ns_str
1034   \str_clear:N \l_stex_module_lang_str
1035   \str_clear:N \l_stex_module_sig_str
1036   \str_clear:N \l_stex_module_creators_str
1037   \str_clear:N \l_stex_module_contributors_str
1038   \str_clear:N \l_stex_module_meta_str
1039   \str_clear:N \l_stex_module_srccite_str
1040   \keys_set:nn { stex / module } { #1 }
1041 }

```

```

1042
1043 % module parameters here? In the body?
1044

```

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

```

1045 \cs_new_protected:Nn \stex_module_setup:nn {
1046   \str_set:Nx \l_stex_module_name_str { #2 }
1047   \__stex_modules_args:n { #1 }

```

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

```

1048   \stex_if_in_module:TF {
1049     % Nested module
1050     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1051       { ns } \l_stex_module_ns_str
1052     \str_set:Nx \l_stex_module_name_str {
1053       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1054         { name } / \l_stex_module_name_str
1055     }
1056   }{
1057     % not nested:
1058     \str_if_empty:NT \l_stex_module_ns_str {
1059       \stex_modules_current_namespace:
1060       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1061       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1062         / { \l_stex_module_ns_str }
1063       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1064       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1065         \str_set:Nx \l_stex_module_ns_str {
1066           \stex_path_to_string:N \l_tmpa_seq
1067         }
1068       }
1069     }
1070   }

```

Next, we determine the language of the module:

```

1071   \str_if_empty:NT \l_stex_module_lang_str {
1072     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1073     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1074     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1075     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1076     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1077       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1078         inferred~from~file~name}
1079       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1080     }
1081   }
1082
1083   \str_if_empty:NF \l_stex_module_lang_str {
1084     \prop_get:NVNTF {c_stex_languages_prop \l_stex_module_lang_str
1085       \l_tmpa_str {
1086       \ltx@ifpackageloaded{babel}{
1087         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1088       }{}

```

```

1089     } {
1090         \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1091     }
1092 }

```

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

```

1093 \str_if_empty:NTF \l_stex_module_sig_str {
1094     \exp_args:Nnx \prop_gset_from_keyval:cn {
1095         c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1096     } {
1097         name      = \l_stex_module_name_str ,
1098         ns        = \l_stex_module_ns_str ,
1099         file      = \exp_not:o { \g_stex_currentfile_seq } ,
1100         lang      = \l_stex_module_lang_str ,
1101         sig       = \l_stex_module_sig_str ,
1102         meta      = \l_stex_module_meta_str
1103     }
1104     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1105     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1106     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1107     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1108     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1109 \str_if_empty:NT \l_stex_module_meta_str {
1110     \str_set:Nx \l_stex_module_meta_str {
1111         \c_stex_metatheory_ns_str ? Metatheory
1112     }
1113 }
1114 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1115     \bool_set_true:N \l_stex_in_meta_bool
1116     \exp_args:Nx \stex_add_to_current_module:n {
1117         \bool_set_true:N \l_stex_in_meta_bool
1118         \stex_activate_module:n {\l_stex_module_meta_str}
1119         \bool_set_false:N \l_stex_in_meta_bool
1120     }
1121     \stex_activate_module:n {\l_stex_module_meta_str}
1122     \bool_set_false:N \l_stex_in_meta_bool
1123 }
1124 }{
1125     \str_if_empty:NT \l_stex_module_lang_str {
1126         \msg_error:nnxx{stex}{error/siglanguage}{
1127             \l_stex_module_ns_str?\l_stex_module_name_str
1128         }\l_stex_module_sig_str}
1129     }
1130
1131     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1132     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1133     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1134     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1135     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1136     \str_set:Nx \l_tmpa_str {
1137         \stex_path_to_string:N \l_tmpa_seq /

```

```

1138     \l_tmpa_str . \l_stex_module_sig_str .tex
1139 }
1140 \IfFileExists \l_tmpa_str {
1141     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1142         \seq_clear:N \l_stex_all_modules_seq
1143         %\prop_clear:N \l_stex_current_module_prop
1144         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1145         \input { \l_tmpa_str }
1146     }
1147 }{
1148     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1149 }
1150 \stex_activate_module:n {
1151     \l_stex_module_ns_str ? \l_stex_module_name_str
1152 }
1153 %\prop_set_eq:Nc \l_stex_current_module_prop {
1154 %   c_stex_module_
1155 %   \l_stex_module_ns_str ?
1156 %   \l_stex_module_name_str
1157 %   _prop
1158 %}
1159 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1160 }
1161 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 25.)

module The module environment.

```

\__stex_modules_begin_module:nn implements \begin{module}

1162 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1163     \stex_reactivate_macro:N \STEXexport
1164     \stex_reactivate_macro:N \importmodule
1165     \stex_reactivate_macro:N \symdecl
1166     \stex_reactivate_macro:N \notation
1167     \stex_reactivate_macro:N \symdef
1168     \stex_module_setup:nn{#1}{#2}
1169 }
1170 \stex_debug:nn{modules}{
1171     New~module:\\
1172     Namespace:~\l_stex_module_ns_str\\
1173     Name:~\l_stex_module_name_str\\
1174     Language:~\l_stex_module_lang_str\\
1175     Signature:~\l_stex_module_sig_str\\
1176     Metatheory:~\l_stex_module_meta_str\\
1177     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1178 }
1179 }
1180 \seq_put_right:Nx \l_stex_all_modules_seq {
1181     \l_stex_module_ns_str ? \l_stex_module_name_str
1182 }
1183 }
1184 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1185 %     { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1186
1187 \stex_if_smsmode:TF {
1188   \stex_smsmode_set_codes:
1189 } {
1190   \begin{stex_annotate_env} {theory} {
1191     \l_stex_module_ns_str ? \l_stex_module_name_str
1192   }
1193
1194   \stex_annotate_invisible:nnn{header}{} {
1195     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1196     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1197     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1198       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1199     }
1200   }
1201 }
1202 % TODO: Inherit metatheory for nested modules?
1203 }
1204 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:nn`.)

`_stex_modules_end_module:` implements `\end{module}`

```

1205 \cs_new_protected:Nn \_stex_modules_end_module: {
1206 % \str_set:Nx \l_tmpa_str {
1207 %   c_stex_module_
1208 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1209 %   \prop_item:Nn \l_stex_current_module_prop { name }
1210 %   _prop
1211 % }
1212 %^^A \prop_new:c { \l_tmpa_str }
1213 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1214 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1215 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1216 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1217 \NewDocumentEnvironment { @module } { 0{} m } {
1218   \par
1219   \_stex_modules_begin_module:nn{#1}{#2}
1220 } {
1221   \_stex_modules_end_module:
1222   \stex_if_smsmode:TF {
1223 %     \exp_args:Nx \stex_add_to_sms:n {
1224 %       \prop_gset_from_keyval:cn {
1225 %         c_stex_module_
1226 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1227 %         \prop_item:Nn \l_stex_current_module_prop { name }
1228 %         _prop
1229 %       } {
1230 %         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1231 %         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,

```



```

1232 %         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1233 %         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1234 %         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1235 %         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1236 %     }
1237 % }
1238 }{
1239   \end{stex_annotate_env}
1240 }
1241 }

```

\stex_modules_heading: Code for document headers

```

1242 \cs_if_exist:NTF \thesection {
1243   \newcounter{module}[section]
1244 }{
1245   \newcounter{module}
1246 }
1247
1248 \bool_if:NT \c_stex_showmods_bool {
1249   \latexml_if:F { \RequirePackage{mdframed} }
1250 }
1251
1252 \cs_new_protected:Nn \stex_modules_heading: {
1253   \stepcounter{module}
1254   \par
1255   \bool_if:NT \c_stex_showmods_bool {
1256     \noindent{\textbf{Module} ~
1257       \cs_if_exist:NT \thesection {\thesection.}
1258       \themodule ~ [\l_stex_module_name_str]
1259     }
1260     \str_if_empty:NTF \l_stex_module_title_str {
1261       }{
1262         \quad(\l_stex_module_title_str)\hfill
1263       }\par
1264     }
1265     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1266     % TODO
1267     \stex_ref_new_doc_target:n \l_stex_module_name_str
1268   }

```

(End definition for \stex_modules_heading:. This function is documented on page 25.)

Finally:

```

1269 \NewDocumentEnvironment { module } { 0 } { m } {
1270   \bool_if:NT \c_stex_showmods_bool {
1271     \begin{mdframed}
1272   }
1273   \begin{@module}[#1]{#2}
1274     \stex_modules_heading:
1275   }{
1276     \end{@module}
1277     \bool_if:NT \c_stex_showmods_bool {
1278       \end{mdframed}
1279     }
1280   }

```

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1281 \NewDocumentCommand \STEXModule { m } {
1282   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1283   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1284   \tl_set:Nn \l_tmpa_tl {
1285     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1286   }
1287   \seq_map_inline:Nn \l_stex_all_modules_seq {
1288     \str_set:Nn \l_tmpb_str { ##1 }
1289     \str_if_eq:eeT { \l_tmpa_str } {
1290       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1291     } {
1292       \seq_map_break:n {
1293         \tl_set:Nn \l_tmpa_tl {
1294           \stex_invoke_module:n { ##1 }
1295         }
1296       }
1297     }
1298   }
1299   \l_tmpa_tl
1300 }
1301
1302 \cs_new_protected:Nn \stex_invoke_module:n {
1303   \stex_debug:nn{modules}{Invoking~module~#1}
1304   \peek_charcode_remove:NTF ! {
1305     \__stex_modules_invoke_uri:nN { #1 }
1306   } {
1307     \peek_charcode_remove:NTF ? {
1308       \__stex_modules_invoke_symbol:nn { #1 }
1309     } {
1310       \msg_error:nnx{stex}{error/syntax}{
1311         ?~or~!~expected~after~
1312         \c_backslash_str STEXModule{#1}
1313       }
1314     }
1315   }
1316 }
1317
1318 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1319   \str_set:Nn #2 { #1 }
1320 }
1321
1322 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1323   \stex_invoke_symbol:n{#1?#2}
1324 }

```

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

```

\stex_activate_module:n
1325 \bool_new:N \l_stex_in_meta_bool
1326 \bool_set_false:N \l_stex_in_meta_bool

```

```

1327 \cs_new_protected:Nn \stex_activate_module:n {
1328   \stex_debug:nn{modules}{Activating~module~#1}
1329   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1330     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1331   }
1332   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1333     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1334     \use:c{ c_stex_module_#1_code }
1335   }
1336 }

```

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

```

1337 </package>

```

Chapter 28

sTeX -Module Inheritance Implementation

```
1338 <*package>
1339
1340 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1341
```

28.1 SMS Mode

```
1342 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1343 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1344 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1345 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1346
1347 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1348   \makeatletter
1349   \makeatother
1350   \ExplSyntaxOn
1351   \ExplSyntaxOff
1352 }
1353
1354 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1355   \symdef
1356   \importmodule
1357   \notation
1358   \symdecl
1359   \STEXexport
1360 }
1361
1362 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1363   \tl_to_str:n {
1364     module,
1365     @module
```

```

1366 }
1367 }

```

(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 28.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
1368 \bool_new:N \g__stex_smsmode_bool
1369 \bool_set_false:N \g__stex_smsmode_bool
1370 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1371   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1372 }

```

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

```

\__stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\__stex_smsmode_if_catcodes:TF
1373 \bool_new:N \g__stex_smsmode_catcode_bool
1374 \bool_set_false:N \g__stex_smsmode_catcode_bool
1375 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1376   \bool_if:NTF \g__stex_smsmode_catcode_bool
1377   \prg_return_true: \prg_return_false:
1378 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:
1379 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1380   \stex_if_smsmode:T {
1381     \__stex_smsmode_if_catcodes:F {
1382       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1383       \exp_after:wN \char_gset_active_eq:NN
1384       \c_backslash_str \__stex_smsmode_cs:
1385       \tex_global:D \char_set_catcode_active:N \
1386       \tex_global:D \char_set_catcode_other:N $
1387       \tex_global:D \char_set_catcode_other:N ^
1388       \tex_global:D \char_set_catcode_other:N _
1389       \tex_global:D \char_set_catcode_other:N &
1390       \tex_global:D \char_set_catcode_other:N ##
1391     }
1392   }
1393 } \iffalse $ \fi % to make syntax highlighting work again

```

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

```

\__stex_smsmode_unset_codes: Sets category code scheme back from the one used in SMS mode.
1394 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1395   \__stex_smsmode_if_catcodes:T {
1396     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1397     \exp_after:wN \tex_global:D \exp_after:wN
1398     \char_set_catcode_escape:N \c_backslash_str
1399     \tex_global:D \char_set_catcode_math_toggle:N $
1400     \tex_global:D \char_set_catcode_math_superscript:N ^
1401     \tex_global:D \char_set_catcode_math_subscript:N _
1402     \tex_global:D \char_set_catcode_alignment:N &
1403     \tex_global:D \char_set_catcode_parameter:N ##
1404   }
1405 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1406 \cs_new_protected:Nn \stex_in_smsmode:nn {
1407   \vbox_set:Nn \l_tmpa_box {
1408     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1409     \bool_gset_true:N \g__stex_smsmode_bool
1410     \stex_smsmode_set_codes:
1411     #2
1412     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1413     \stex_if_smsmode:F {
1414       \__stex_smsmode_unset_codes:
1415     }
1416   }
1417   \box_clear:N \l_tmpa_box
1418 }

```

(End definition for `\stex_in_smsmode:nn`. This function is documented on page [29](#).)

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

```

1419 \cs_new_protected:Nn \_stex_smsmode_cs: {
1420   \str_clear:N \l_tmpa_str
1421   \peek_analysis_map_inline:n {
1422     % #1: token (one expansion)
1423     % #2: charcode
1424     % #3 catcode
1425     \token_if_eq_charcode:NNTF ##3 B {
1426       % token is a letter
1427       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1428     } {
1429       \str_if_empty:NNTF \l_tmpa_str {
1430         % we don't allow (or need) single non-letter CSs
1431         % for now
1432         \peek_analysis_map_break:
1433       }{
1434         \str_if_eq:onTF \l_tmpa_str { begin } {
1435           \peek_analysis_map_break:n {
1436             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1437           }
1438         } {
1439           \str_if_eq:onTF \l_tmpa_str { end } {
1440             \peek_analysis_map_break:n {
1441               \exp_after:wN \_stex_smsmode_checkend:n ##1
1442             }
1443           } {
1444             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1445             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1446             \g_stex_smsmode_allowedmacros_tl
1447             { \use:c{\l_tmpa_str} } {
1448               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1449               \peek_analysis_map_break:n {
1450                 \exp_after:wN \l_tmpa_tl ##1
1451               }

```

```

1452     } {
1453         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1454         \g_stex_smsmode_allowedmacros_escape_tl
1455         { \use:c{\l_tmpa_str} } {
1456             \__stex_smsmode_unset_codes:
1457             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1458             % TODO \__stex_smsmode_rescan_cs:
1459             % \int_compare:nNnTF {##2} = {92} {
1460             %     \peek_analysis_map_break:n {
1461             %         \__stex_smsmode_unset_codes:
1462             %         \__stex_smsmode_rescan_cs:
1463             %     }
1464             % } {
1465             %     \peek_analysis_map_break:n {
1466             %         \exp_after:wN \l_tmpa_tl ##1
1467             %     }
1468             % }
1469             } {
1470                 \int_compare:nNnTF {##2} = {92} {
1471                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1472                 }{
1473                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1474                 }
1475             }
1476         }
1477     }
1478 }
1479 }
1480 }
1481 }
1482 }

```

(End definition for __stex_smsmode_cs:.)

__stex_smsmode_rescan_cs: If the last token gobbled by \stex_smsmode_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```

1483 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1484     \str_clear:N \l_tmpb_str
1485     \peek_analysis_map_inline:n {
1486         \token_if_eq_charcode:NNTF ##3 B {
1487             % token is a letter
1488             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1489         } {
1490             \peek_analysis_map_break:n {
1491                 \exp_after:wN \use:c \exp_after:wN {
1492                     \exp_after:wN \l_tmpa_str\exp_after:wN
1493                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1494             }
1495         }
1496     }
1497 }

```

(End definition for __stex_smsmode_rescan_cs:.)

`__stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1498 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1499   \str_set:Nn \l_tmpa_str { #1 }
1500   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1501     \__stex_smsmode_unset_codes:
1502     \begin{#1}
1503   }
1504 }
```

(End definition for `__stex_smsmode_checkbegin:n`.)

`__stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1505 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1506   \str_set:Nn \l_tmpa_str { #1 }
1507   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1508     \end{#1}
1509   }
1510 }
```

(End definition for `__stex_smsmode_checkend:n`.)

28.2 Inheritance

```

1511 <@@=stex_importmodule>
```

`\stex_import_module_uri:nn`

```

1512 \cs_new_protected:Nn \stex_import_module_uri:nn {
1513   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1514   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1515
1516   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1517   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1518   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1519
1520   \stex_modules_current_namespace:
1521   \bool_lazy_all:nTF {
1522     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1523     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1524     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1525   }{
1526     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1527     \str_set_eq:NN \l_stex_module_ns
1528   }{
1529     \str_if_empty:NT \l__stex_importmodule_archive_str {
1530       \prop_if_empty:NF \l_stex_current_repository_prop {
1531         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1532       }
1533     }
1534     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1535       \str_if_empty:NF \l__stex_importmodule_path_str {
1536         \str_set:Nx \l_stex_module_ns_str {
1537           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1538         }
1539       }
```



```

1540   }{
1541     \stex_require_repository:n \l__stex_importmodule_archive_str
1542     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1543       \l_stex_module_ns_str
1544     \str_if_empty:NF \l__stex_importmodule_path_str {
1545       \str_set:Nx \l_stex_module_ns_str {
1546         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1547       }
1548     }
1549   }
1550 }
1551 }

```

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

<code>\l__stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l__stex_importmodule_archive_str</code>	1552 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1553 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1554 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1555 <code>\str_new:N \g__stex_importmodule_file_str</code>

(End definition for `\l__stex_importmodule_name_str` and others.)

```

\stex_import_require_module:nnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1556 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1557   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1558
1559     % archive
1560     \str_set:Nx \l_tmpa_str { #2 }
1561     \str_if_empty:NTF \l_tmpa_str {
1562       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1563     } {
1564       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1565       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1566       \seq_put_right:Nn \l_tmpa_seq { source }
1567     }
1568
1569     % path
1570     \str_set:Nx \l_tmpb_str { #3 }
1571     \str_if_empty:NTF \l_tmpb_str {
1572       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1573
1574       \ltx@ifpackageloaded{babel} {
1575         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1576           { \language } \l_tmpb_str {
1577           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1578         }
1579       } {
1580         \str_clear:N \l_tmpb_str
1581       }
1582
1583       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1584       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1585         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1586     }{
1587       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1588       \IfFileExists{ \l_tmpa_str.tex }{
1589         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1590       }{
1591         % try english as default
1592         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1593         \IfFileExists{ \l_tmpa_str.en.tex }{
1594           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1595         }{
1596           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1597         }
1598       }
1599     }
1600
1601   } {
1602     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1603     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1604
1605     \ltx@ifpackageloaded{babel} {
1606       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1607         { \language } \l_tmpb_str {
1608         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1609       }
1610     } {
1611       \str_clear:N \l_tmpb_str
1612     }
1613
1614     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1615
1616     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1617     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1618       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1619     }{
1620       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1621       \IfFileExists{ \l_tmpa_str/#4.tex }{
1622         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1623       }{
1624         % try english as default
1625         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1626         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1627           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1628         }{
1629           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1630           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1631             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1632           }{
1633             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1634             \IfFileExists{ \l_tmpa_str.tex }{
1635               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1636             }{
1637               % try english as default
1638               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1639               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1640         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1641     }{
1642         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1643     }
1644 }
1645 }
1646 }
1647 }
1648 }
1649 }
1650
1651 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1652     \seq_clear:N \l_stex_all_modules_seq
1653     \str_clear:N \l_stex_current_module_str
1654     \str_set:Nx \l_tmpb_str { #2 }
1655     \str_if_empty:NF \l_tmpb_str {
1656         \stex_set_current_repository:n { #2 }
1657     }
1658     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1659     \input { \g__stex_importmodule_file_str }
1660 }
1661
1662 \stex_if_module_exists:nF { #1 ? #4 } {
1663     \msg_error:nnx{stex}{error/unknownmodule}{
1664         #1?#4~(in~file~\g__stex_importmodule_file_str)
1665     }
1666 }
1667 }
1668 \stex_activate_module:n { #1 ? #4 }
1669 }

```

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

`\importmodule`

```

1670 \NewDocumentCommand \importmodule { 0{} m } {
1671     \stex_import_module_uri:nn { #1 } { #2 }
1672     \stex_debug:nn{modules}{Importing~module:~
1673         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1674     }
1675     \stex_if_smsmode:F {
1676         \stex_import_require_module:nnnn
1677         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1678         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1679         \stex_annotate_invisible:nnn
1680         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1681     }
1682     \exp_args:Nx \stex_add_to_current_module:n {
1683         \stex_import_require_module:nnnn
1684         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1685         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1686     }
1687     \exp_args:Nx \stex_add_import_to_current_module:n {
1688         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1689     }

```

```

1690 \stex_smsmode_set_codes:
1691 }
1692 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

(End definition for \importmodule. This function is documented on page 29.)

\usemodule

```

1693 \NewDocumentCommand \usemodule { 0{} m } {
1694   \stex_if_smsmode:F {
1695     \stex_import_module_uri:nn { #1 } { #2 }
1696     \stex_import_require_module:nnnn
1697     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1698     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1699     \stex_annotate_invisible:nnn
1700     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1701   }
1702   \stex_smsmode_set_codes:
1703 }

```

(End definition for \usemodule. This function is documented on page 30.)

```

1704 \endpackage

```

Chapter 29

STEX -Symbols Implementation

```
1705 <*package>
1706
1707 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1708
```

Warnings and error messages

```
1709
```

29.1 Symbol Declarations

```
1710 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1711 \seq_new:N \l_stex_all_symbols_seq
```

(End definition for `\l_stex_all_symbols_seq`. This variable is documented on page 33.)

`\STEXsymbol`

```
1717 \NewDocumentCommand \STEXsymbol { m } {
1718   \stex_get_symbol:n { #1 }
1719   \exp_args:No
1715   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1716 }
```

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

symdecl arguments:

```
1717 \keys_define:nn { stex / symdecl } {
1718   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1719   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1720   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1721   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1722   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1723   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1724   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1725   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1726 }
```

```

1727
1728 \bool_new:N \l_stex_symdecl_make_macro_bool
1729
1730 \cs_new_protected:Nn \__stex_symdecl_args:n {
1731   \str_clear:N \l_stex_symdecl_name_str
1732   \str_clear:N \l_stex_symdecl_args_str
1733   \bool_set_false:N \l_stex_symdecl_local_bool
1734   \tl_clear:N \l_stex_symdecl_type_tl
1735   \tl_clear:N \l_stex_symdecl_definiens_tl
1736
1737   \keys_set:nn { stex / symdecl } { #1 }
1738 }

```

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

```

1739
1740 \NewDocumentCommand \symdecl { s O{} m } {
1741   \__stex_symdecl_args:n { #2 }
1742   \IfBooleanTF #1 {
1743     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1744   } {
1745     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1746   }
1747   \stex_symdecl_do:n { #3 }
1748   \stex_smsmode_set_codes:
1749 }
1750 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1751 \cs_new_protected:Nn \stex_symdecl_do:n {
1752   \stex_if_in_module:F {
1753     % TODO throw error? some default namespace?
1754   }
1755
1756   \str_if_empty:NT \l_stex_symdecl_name_str {
1757     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1758   }
1759
1760   \prop_if_exist:cT { l_stex_symdecl_
1761     \l_stex_current_module_str ?
1762     \l_stex_symdecl_name_str
1763     _prop
1764   }{
1765     % TODO throw error (beware of circular dependencies)
1766   }
1767
1768   \prop_clear:N \l_tmpa_prop
1769   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1770   \seq_clear:N \l_tmpa_seq
1771   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1772   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1773

```

```

1774 \exp_args:No \stex_add_constant_to_current_module:n {
1775   \l_stex_symdecl_name_str
1776 }
1777
1778 % arity/args
1779 \int_zero:N \l_tmpb_int
1780
1781 \bool_set_true:N \l_tmpa_bool
1782 \str_map_inline:Nn \l_stex_symdecl_args_str {
1783   \token_case_meaning:NnF ##1 {
1784     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1785     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1786     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1787     {\tl_to_str:n a} {
1788       \bool_set_false:N \l_tmpa_bool
1789       \int_incr:N \l_tmpb_int
1790     }
1791     {\tl_to_str:n B} {
1792       \bool_set_false:N \l_tmpa_bool
1793       \int_incr:N \l_tmpb_int
1794     }
1795   }{
1796     \msg_set:nnn{stex}{error/wrongargs}{
1797       args~value~in~symbol~declaration~for~
1798       \l_stex_current_module_str ?
1799       \l_stex_symdecl_name_str ~
1800       needs~to~be~
1801       i,~a,~b~or~B,~but~##1~given
1802     }
1803     \msg_error:nn{stex}{error/wrongargs}
1804   }
1805 }
1806 \bool_if:NTF \l_tmpa_bool {
1807   % possibly numeric
1808   \str_if_empty:NTF \l_stex_symdecl_args_str {
1809     \prop_put:Nnn \l_tmpa_prop { args } {}
1810     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1811   }{
1812     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1813     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1814     \str_clear:N \l_tmpa_str
1815     \int_step_inline:nn \l_tmpa_int {
1816       \str_put_right:Nn \l_tmpa_str i
1817     }
1818     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1819   }
1820 } {
1821   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1822   \prop_put:Nnx \l_tmpa_prop { arity }
1823     { \str_count:N \l_stex_symdecl_args_str }
1824 }
1825 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1826
1827

```

```

1828 % semantic macro
1829
1830 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1831   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1832     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1833   } }
1834
1835   \bool_if:NF \l_stex_symdecl_local_bool {
1836     \exp_args:Nx \stex_add_to_current_module:n {
1837       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1838         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1839       } }
1840     }
1841   }
1842 }
1843
1844 % add to all symbols
1845
1846 \bool_if:NF \l_stex_symdecl_local_bool {
1847   \exp_args:Nx \stex_add_to_current_module:n {
1848     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1849       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1850     }
1851   }
1852   \exp_args:Nx \stex_add_field_to_current_module:n {
1853     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1854   }
1855 }
1856
1857 \stex_debug:nn{symbols}{New~symbol:~
1858   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1859   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1860   Args:~\prop_item:Nn \l_tmpa_prop { args }
1861 }
1862
1863 % circular dependencies require this:
1864
1865 \prop_if_exist:cF {
1866   l_stex_symdecl_
1867   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1868   _prop
1869 } {
1870   \prop_set_eq:cN {
1871     l_stex_symdecl_
1872     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1873     _prop
1874   } \l_tmpa_prop
1875 }
1876
1877 \seq_clear:c {
1878   l_stex_symdecl_
1879   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1880   _notations
1881 }

```



```

1882
1883 \bool_if:NF \l_stex_symdecl_local_bool {
1884   \exp_args:Nx
1885   \stex_add_to_current_module:n {
1886     \seq_clear:c {
1887       l_stex_symdecl_
1888       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1889       _notations
1890     }
1891     \prop_set_from_keyval:cn {
1892       l_stex_symdecl_
1893       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1894       _prop
1895     } {
1896       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1897       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1898       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1899       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1900       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1901       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1902     }
1903   }
1904 }
1905
1906 \stex_if_smsmode:TF {
1907   \bool_if:NF \l_stex_symdecl_local_bool {
1908     % \exp_args:Nx \stex_add_to_sms:n {
1909     %   \prop_set_from_keyval:cn {
1910     %     l_stex_symdecl_
1911     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1912     %     _prop
1913     %   } {
1914     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1915     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1916     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1917     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1918     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1919     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1920     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1921     %   }
1922     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1923     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1924     %   }
1925     % }
1926   }
1927 }{
1928   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1929     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1930   }
1931   \stex_if_do_html:T {
1932     \stex_annotate_invisible:nnn {symdecl} {
1933       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1934     } {
1935       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st

```

```

1936     \stex_annotate_invisible:nnn{args}{}{
1937     \prop_item:Nn \l_tmpa_prop { args }
1938   }
1939   \stex_annotate_invisible:nnn{macroname}{}{#1}
1940   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1941     \stex_annotate_invisible:nnn{definiens}{}
1942     { $\l_stex_symdecl_definiens_tl$ }
1943   }
1944 }
1945 }
1946 }
1947 }

```

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

`\stex_get_symbol:n`

```

1948 \str_new:N \l_stex_get_symbol_uri_str
1949
1950 \cs_new_protected:Nn \stex_get_symbol:n {
1951   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1952     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1953   }{
1954     % argument is a string
1955     % is it a command name?
1956     \cs_if_exist:cTF { #1 }{
1957       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1958       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1959       \str_if_empty:NNTF \l_tmpa_str {
1960         \exp_args:Nx \cs_if_eq:NNTF {
1961           \tl_head:N \l_tmpa_tl
1962         } \stex_invoke_symbol:n {
1963           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1964         }{
1965           \__stex_symdecl_get_symbol_from_string:n { #1 }
1966         }
1967       } {
1968         \__stex_symdecl_get_symbol_from_string:n { #1 }
1969       }
1970     }{
1971       % argument is not a command name
1972       \__stex_symdecl_get_symbol_from_string:n { #1 }
1973       % \l_stex_all_symbols_seq
1974     }
1975   }
1976 }
1977
1978 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1979   \str_set:Nn \l_tmpa_str { #1 }
1980   \bool_set_false:N \l_tmpa_bool
1981   \stex_if_in_module:T {
1982     \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
1983       \bool_set_true:N \l_tmpa_bool
1984       \str_set:Nx \l_stex_get_symbol_uri_str {
1985         \l_stex_current_module_str ? #1

```

```

1986     }
1987   }
1988 }
1989 \bool_if:NF \l_tmpa_bool {
1990   \tl_set:Nn \l_tmpa_tl {
1991     \msg_set:nnn{stex}{error/unknownsymbol}{
1992       No~symbol~#1~found!
1993     }
1994     \msg_error:nn{stex}{error/unknownsymbol}
1995   }
1996   \str_set:Nn \l_tmpa_str { #1 }
1997   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1998   \seq_map_inline:Nn \l_stex_all_symbols_seq {
1999     \str_set:Nn \l_tmpb_str { ##1 }
2000     \str_if_eq:eeT { \l_tmpa_str } {
2001       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2002     } {
2003       \seq_map_break:n {
2004         \tl_set:Nn \l_tmpa_tl {
2005           \str_set:Nn \l_stex_get_symbol_uri_str {
2006             ##1
2007           }
2008         }
2009       }
2010     }
2011   }
2012   \l_tmpa_tl
2013 }
2014 }
2015
2016 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2017   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2018   { \tl_tail:N \l_tmpa_tl }
2019   \tl_if_single:NTF \l_tmpa_tl {
2020     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2021       \exp_after:wN \str_set:Nn \exp_after:wN
2022       \l_stex_get_symbol_uri_str \l_tmpa_tl
2023     }{
2024       % TODO
2025       % tail is not a single group
2026     }
2027   }{
2028     % TODO
2029     % tail is not a single group
2030   }
2031 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page [33](#).)

29.2 Notations

```

2032 <@@=stex_notation>

```

notation arguments:

```

2033 \keys_define:nn { stex / notation } {
2034   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2035   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2036   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2037   op        .tl_set:N   = \l__stex_notation_op_tl ,
2038   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2039   primary   .default:n  = {true} ,
2040   unknown   .code:n     = \str_set:Nx
2041             \l__stex_notation_variant_str \l_keys_key_str
2042 }
2043
2044 \cs_new_protected:Nn \__stex_notation_args:n {
2045   \str_clear:N \l__stex_notation_lang_str
2046   \str_clear:N \l__stex_notation_variant_str
2047   \str_clear:N \l__stex_notation_prec_str
2048   \tl_clear:N \l__stex_notation_op_tl
2049   \bool_set_false:N \l__stex_notation_primary_bool
2050
2051   \keys_set:nn { stex / notation } { #1 }
2052 }

```

\notation

```

2053 \NewDocumentCommand \notation { 0{ } m } {
2054   \__stex_notation_args:n { #1 }
2055   \tl_clear:N \l_stex_symdecl_definiens_tl
2056   \stex_get_symbol:n { #2 }
2057   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2058 }
2059 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2060 \cs_new_protected:Nn \stex_notation_do:nn {
2061   \let\l_stex_current_symbol_str\relax
2062   \prop_set_eq:Nc \l_tmpa_prop {
2063     l_stex_symdecl_ #1 _prop
2064   }
2065
2066   \prop_clear:N \l_tmpb_prop
2067   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2068   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2069   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2070
2071   % precedences
2072   \seq_clear:N \l_tmpb_seq
2073   \exp_args:NNno
2074   \str_if_empty:NTF \l__stex_notation_prec_str {
2075     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2076     \int_compare:nNnTF \l_tmpa_str = 0 {
2077       \exp_args:NNnx
2078       \prop_put:Nno \l_tmpb_prop { opprec }
2079       { \neginfprec }
2080     }{
2081       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }

```

```

2082     }
2083   } {
2084     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2085       \exp_args:NNnx
2086       \prop_put:Nno \l_tmpb_prop { opprec }
2087       { \neginfprec }
2088       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2089       \int_step_inline:nn { \l_tmpa_str } {
2090         \exp_args:NNx
2091         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2092       }
2093     }{
2094       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2095       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2096         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2097         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2098           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2099             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2100           \seq_map_inline:Nn \l_tmpa_seq {
2101             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2102           }
2103         }
2104         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2105       }{
2106         \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2107         \int_compare:nNnTF \l_tmpa_str = 0 {
2108           \exp_args:NNnx
2109           \prop_put:Nno \l_tmpb_prop { opprec }
2110           { \infprec }
2111         }{
2112           \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2113         }
2114       }
2115     }
2116   }
2117
2118   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2119   \int_step_inline:nn { \l_tmpa_str } {
2120     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2121       \exp_args:NNx
2122       \seq_put_right:Nn \l_tmpb_seq {
2123         \prop_item:Nn \l_tmpb_prop { opprec }
2124       }
2125     }
2126   }
2127
2128   \prop_put:Nno \l_tmpb_prop { argprecs } \l_tmpb_seq
2129   \tl_clear:N \l_tmpa_tl
2130
2131   \int_compare:nNnTF \l_tmpa_str = 0 {
2132     \exp_args:NNe
2133     \cs_set:Npn \l__stex_notation_macrocode_cs {
2134       \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2135       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2136         { \prop_item:Nn \l_tmpb_prop { opprec } }
2137         { \exp_not:n { #2 } }
2138     }
2139     \__stex_notation_final:
2140 }{
2141     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2142     \str_if_in:NnTF \l_tmpb_str b {
2143         \exp_args:Nne \use:nn
2144         {
2145             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2146             \cs_set:Npn \l_tmpa_str } { {
2147                 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2148                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2149                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2150                 { \exp_not:n { #2 } }
2151             } }
2152     }{
2153         \str_if_in:NnTF \l_tmpb_str B {
2154             \exp_args:Nne \use:nn
2155             {
2156                 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2157                 \cs_set:Npn \l_tmpa_str } { {
2158                     \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2159                     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2160                     { \prop_item:Nn \l_tmpb_prop { opprec } }
2161                     { \exp_not:n { #2 } }
2162                 } }
2163     }{
2164         \exp_args:Nne \use:nn
2165         {
2166             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2167             \cs_set:Npn \l_tmpa_str } { {
2168                 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2169                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2170                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2171                 { \exp_not:n { #2 } }
2172             } }
2173     }
2174 }
2175
2176 \int_zero:N \l_tmpa_int
2177 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2178 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2179 \__stex_notation_arguments:
2180 }
2181 }

```

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

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

```

2182 \cs_new_protected:Nn \__stex_notation_arguments: {
2183     \int_incr:N \l_tmpa_int
2184     \str_if_empty:NnTF \l_tmpa_str {
2185         \__stex_notation_final:

```

```

2186 }{
2187   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2188   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2189   \str_if_eq:VnTF \l_tmpb_str a {
2190     \__stex_notation_argument_assoc:n
2191   }{
2192     \str_if_eq:VnTF \l_tmpb_str B {
2193       \__stex_notation_argument_assoc:n
2194     }{
2195       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2196       \tl_put_right:Nx \l_tmpa_tl {
2197         { \stex_term_math_arg:nnn
2198           { \int_use:N \l_tmpa_int }
2199           { \l_tmpb_str }
2200           { ####\int_use:N \l_tmpa_int }
2201         }
2202       }
2203       \__stex_notation_arguments:
2204     }
2205   }
2206 }
2207 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2208 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2209   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2210   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2211   \tl_put_right:Nx \l_tmpa_tl {
2212     { \stex_term_math_assoc_arg:nnnn
2213       { \int_use:N \l_tmpa_int }
2214       { \l_tmpb_str }
2215       \exp_args:No \exp_not:n
2216       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2217       { ####\int_use:N \l_tmpa_int }
2218     }
2219   }
2220   \__stex_notation_arguments:
2221 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2222 \cs_new_protected:Nn \__stex_notation_final: {
2223   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2224   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2225   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2226   \exp_args:Nne \use:nn
2227   {
2228     \cs_generate_from_arg_count:cNnn {
2229       stex_notation_ \l_tmpa_str \c_hash_str
2230       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2231       _cs

```

```

2232     }
2233     \cs_set:Npn \l_tmpb_str } { {
2234         \exp_after:wN \exp_after:wN \exp_after:wN
2235         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2236         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2237     } }
2238
2239     \tl_if_empty:NF \l__stex_notation_op_tl {
2240         \cs_set:cpx {
2241             stex_op_notation_ \l_tmpa_str \c_hash_str
2242             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2243             _cs
2244         } {
2245             \stex_term_oms:nnn {
2246                 \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2247                 \l__stex_notation_lang_str
2248             }{
2249                 \l_tmpa_str
2250             }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2251         }
2252     }
2253
2254     \exp_args:Ne
2255     \stex_add_to_current_module:n {
2256         \cs_generate_from_arg_count:cNnn {
2257             stex_notation_ \l_tmpa_str \c_hash_str
2258             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2259             _cs
2260         } \cs_set:Npn {\l_tmpb_str} {
2261             \exp_after:wN \exp_after:wN \exp_after:wN
2262             \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2263             { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2264         }
2265         \tl_if_empty:NF \l__stex_notation_op_tl {
2266             \cs_set:cpn {
2267                 stex_op_notation_ \l_tmpa_str \c_hash_str
2268                 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2269                 _cs
2270             } {
2271                 \stex_term_oms:nnn {
2272                     \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2273                     \l__stex_notation_lang_str
2274                 }{
2275                     \l_tmpa_str
2276                 }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2277             }
2278         }
2279     }
2280
2281     \seq_put_right:cx {
2282         l_stex_symdecl_
2283         \prop_item:Nn \l_tmpb_prop { symbol }
2284         _notations
2285     } {

```



```

2286 \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2287 }
2288
2289 \stex_debug:nn{symbols}{
2290   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2291   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2292   Operator~precedence:~
2293   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2294   Argument~precedences:~
2295   \seq_use:Nn \l_tmpa_seq {,~}^^J
2296   Notation: \cs_meaning:c {
2297     stex_notation_ \l_tmpa_str \c_hash_str
2298     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2299     _cs
2300   }
2301 }
2302
2303 \prop_set_eq:cN {
2304   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2305   \c_hash_str \l__stex_notation_lang_str _prop
2306 } \l_tmpb_prop
2307
2308 \exp_args:Ne
2309 \stex_add_to_current_module:n {
2310   \seq_put_right:cn {
2311     l_stex_symdecl_
2312     \prop_item:Nn \l_tmpb_prop { symbol }
2313     _notations
2314   } {
2315     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2316   }
2317   \prop_set_from_keyval:cn {
2318     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2319     \c_hash_str \l__stex_notation_lang_str _prop
2320   } {
2321     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2322     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2323     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2324     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2325     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2326   }
2327 }
2328
2329 \stex_if_smsmode:TF {
2330   \stex_smsmode_set_codes:
2331   % \exp_args:Nx \stex_add_to_sms:n {
2332   %   \prop_set_from_keyval:cn {
2333   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2334   %     \c_hash_str \l__stex_notation_lang_str _prop
2335   %   } {
2336   %     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2337   %     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2338   %     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2339   %     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,

```

```

2340 %      argprec = \prop_item:Nn \l_tmpb_prop { argprec } ,
2341 %    }
2342 %  }
2343 }{
2344
2345 % HTML annotations
2346 \stex_if_do_html:T {
2347   \stex_annotate_invisible:nnn { notation }
2348   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2349     \stex_annotate_invisible:nnn { notationfragment }
2350     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2351     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2352     \stex_annotate_invisible:nnn { precedence }
2353     { \prop_item:Nn \l_tmpb_prop { opprec };
2354       \seq_use:Nn \l_tmpa_seq { x }
2355     }{}
2356
2357     \int_zero:N \l_tmpa_int
2358     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2359     \tl_clear:N \l_tmpa_tl
2360     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2361       \int_incr:N \l_tmpa_int
2362       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2363       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2364       \str_if_eq:VnTF \l_tmpb_str a {
2365         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2366           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2367           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2368         } }
2369       }{
2370         \str_if_eq:VnTF \l_tmpb_str B {
2371           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2372             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2373             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2374           } }
2375         }{
2376           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2377             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2378           } }
2379         }
2380       }
2381     }
2382     \stex_annotate_invisible:nnn { notationcomp }{}{
2383       $ \exp_args:Nno \use:nn { \use:c {
2384         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
2385         \c_hash_str \l__stex_notation_variant_str
2386         \c_hash_str \l__stex_notation_lang_str _cs
2387       } } { \l_tmpa_tl } $
2388     }
2389   }
2390 }
2391 }
2392 }

```

(End definition for `__stex_notation_final:.`)

\symdef

```
2393 \keys_define:nn { stex / symdef } {
2394   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2395   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2396   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2397   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2398   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2399   op        .tl_set:N    = \l__stex_notation_op_tl ,
2400   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2401   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2402   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2403   unknown   .code:n      = \str_set:Nx
2404             \l__stex_notation_variant_str \l_keys_key_str
2405 }
2406
2407 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2408   \str_clear:N \l_stex_symdecl_name_str
2409   \str_clear:N \l_stex_symdecl_args_str
2410   \bool_set_false:N \l_stex_symdecl_local_bool
2411   \tl_clear:N \l_stex_symdecl_type_tl
2412   \tl_clear:N \l_stex_symdecl_definiens_tl
2413   \str_clear:N \l__stex_notation_lang_str
2414   \str_clear:N \l__stex_notation_variant_str
2415   \str_clear:N \l__stex_notation_prec_str
2416   \tl_clear:N \l__stex_notation_op_tl
2417
2418   \keys_set:nn { stex / symdef } { #1 }
2419 }
2420
2421 \NewDocumentCommand \symdef { 0{} m } {
2422   \__stex_notation_symdef_args:n { #1 }
2423   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2424   \stex_symdecl_do:n { #2 }
2425   \exp_args:Nx \stex_notation_do:nn {
2426     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2427   }
2428 }
2429 \stex_deactivate_macro:Nn \symdef {module~environments}
2430 \end{package}
```

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

Chapter 30

STEX -Terms Implementation

```
2431 <*package>
2432
2433 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2434
2435 <@@=stex_terms>
2436
2437 Warnings and error messages
2438 \msg_new:nnn{stex}{error/nonotation}{
2439   Symbol~#1~invoked,~but~has~no~notation~#2!
2440 }
2441 \msg_new:nnn{stex}{error/notationarg}{
2442   Error~in~parsing~notation~#1
2443 }
2444 \msg_new:nnn{stex}{error/noop}{
2445   Symbol~#1~has~no~operator~notation~for~notation~#2
2446 }
```

30.1 Symbol Invocations

Arguments:

```
2446 \keys_define:nn { stex / terms } {
2447   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2448   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2449   unknown .code:n = \str_set:Nx
2450     \l__stex_terms_variant_str \l_keys_key_str
2451 }
2452
2453 \cs_new_protected:Nn \__stex_terms_args:n {
2454   \str_clear:N \l__stex_terms_lang_str
2455   \str_clear:N \l__stex_terms_variant_str
2456   \str_clear:N \l__stex_terms_prec_str
2457   \tl_clear:N \l__stex_terms_op_tl
2458 }
2459 \keys_set:nn { stex / terms } { #1 }
```

2460 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2461 \cs_new_protected:Nn \stex_invoke_symbol:n {
2462   \if_mode_math:
2463     \exp_after:wN \__stex_terms_invoke_math:n
2464   \else:
2465     \exp_after:wN \__stex_terms_invoke_text:n
2466   \fi: { #1 }
2467 }
```

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

__stex_terms_invoke_math:n

```
2468 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2469   \peek_charcode_remove:NTF ! {
2470     \peek_charcode:NTF [ {
2471       \__stex_terms_invoke_op:nw { #1 }
2472     }{
2473       \peek_charcode_remove:NTF ! {
2474         \peek_charcode:NTF [ {
2475           \__stex_terms_invoke_op_custom:nw
2476         }{
2477           % TODO throw error
2478         }
2479       }{
2480         \__stex_terms_invoke_op:nw { #1 } []
2481       }
2482     }{
2483     }{
2484       \peek_charcode_remove:NTF * {
2485         \__stex_terms_invoke_text:n { #1 }
2486       }{
2487         \peek_charcode:NTF [ {
2488           \__stex_terms_invoke_math:nw { #1 }
2489         }{
2490           \__stex_terms_invoke_math:nw { #1 } []
2491         }
2492       }
2493     }
2494 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2495 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2496   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2497     \stex_highlight_term:nn{#1}{#2}
2498   }
2499 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

__stex_terms_invoke_op:nw

```

2500 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2501   \__stex_terms_args:n { #2 }
2502   \cs_if_exist:cTF {
2503     stex_op_notation_ #1 \c_hash_str
2504     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2505   }{
2506     \csname stex_op_notation_ #1 \c_hash_str
2507       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2508     \endcsname
2509   }{
2510     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2511   }
2512 }

```

(End definition for __stex_terms_invoke_op:nw.)

__stex_terms_invoke_math:nw

```

2513 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2514   \__stex_terms_args:n { #2 }
2515   \seq_if_empty:cTF {
2516     l_stex_symdecl_ #1 _notations
2517   } {
2518     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2519   } {
2520     \seq_if_in:cxTF {
2521       l_stex_symdecl_ #1 _notations
2522     }
2523     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2524       \str_set:Nn \l_stex_current_symbol_str { #1 }
2525       \use:c{
2526         stex_notation_ #1 \c_hash_str
2527         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2528         _cs
2529       }
2530     }{
2531       \str_if_empty:NTF \l__stex_terms_variant_str {
2532         \str_if_empty:NTF \l__stex_terms_lang_str {
2533           \seq_get_left:cN {
2534             l_stex_symdecl_ #1 _notations
2535           } \l_tmpa_str
2536           \str_set:Nn \l_stex_current_symbol_str { #1 }
2537           \use:c{
2538             stex_notation_ #1 \c_hash_str \l_tmpa_str
2539             _cs
2540           }
2541         }{
2542           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2543             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2544           }
2545         }
2546       }{
2547         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2548           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2549     }
2550   }
2551 }
2552 }
2553 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2554 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2555   \peek_charcode_remove:NTF ! {
2556     \stex_term_custom:nn { #1 } { }
2557   }{
2558     \prop_set_eq:Nc \l_tmpa_prop {
2559       l_stex_symdecl_ #1 _prop
2560     }
2561     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2562     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2563   }
2564 }

```

(End definition for `_stex_terms_invoke_text:n`.)

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2565 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2566 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2567 \int_new:N \l__stex_terms_downprec
2568 \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 36.)

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2569 \tl_set:Nn \l__stex_terms_left_bracket_str (
2570 \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

```

2571 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2572   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2573     \bool_set_false:N \l__stex_terms_brackets_done_bool
2574     #2
2575   } {
2576     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2577       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2578         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2579         \dobrackets { #2 }
2580       }

```

```

2581     }{ #2 }
2582   }
2583 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2584 \bool_new:N \l__stex_terms_brackets_done_bool
2585 %\RequirePackage{scalerel}
2586 \cs_new_protected:Npn \dobrackets #1 {
2587   %\ThisStyle{\if D\m@switch
2588   %   \exp_args:Nnx \use:nn
2589   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2590   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2591   % \else
2592   \exp_args:Nnx \use:nn
2593   {
2594     \bool_set_true:N \l__stex_terms_brackets_done_bool
2595     \int_set:Nn \l__stex_terms_downprec \infprec
2596     \l__stex_terms_left_bracket_str
2597     #1
2598   }
2599   {
2600     \bool_set_false:N \l__stex_terms_brackets_done_bool
2601     \l__stex_terms_right_bracket_str
2602     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2603   }
2604   %\fi}
2605 }

```

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

`\withbrackets`

```

2606 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2607   \exp_args:Nnx \use:nn
2608   {
2609     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2610     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2611     #3
2612   }
2613   {
2614     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2615     {\l__stex_terms_left_bracket_str}
2616     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2617     {\l__stex_terms_right_bracket_str}
2618   }
2619 }

```

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

`\STEXinvisible`

```

2620 \cs_new_protected:Npn \STEXinvisible #1 {
2621   \stex_annotate_invisible:n { #1 }
2622 }

```


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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2623 \cs_new_protected:Nn \_stex_term_oms:nnn {
2624   \stex_annotate:nnn{ OMID }{ #2 }{
2625     \stex_highlight_term:nn { #1 } { #3 }
2626   }
2627 }
2628
2629 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2630   \__stex_terms_maybe_brackets:nn { #3 }{
2631     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2632   }
2633 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 35.)

`_stex_term_math_oma:nnnn`

```

2634 \cs_new_protected:Nn \_stex_term_oma:nnn {
2635   \stex_annotate:nnn{ OMA }{ #2 }{
2636     \stex_highlight_term:nn { #1 } { #3 }
2637   }
2638 }
2639
2640 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2641   \__stex_terms_maybe_brackets:nn { #3 }{
2642     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2643   }
2644 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 35.)

`_stex_term_math_omb:nnnn`

```

2645 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2646   \stex_annotate:nnn{ OMBIND }{ #2 }{
2647     \stex_highlight_term:nn { #1 } { #3 }
2648   }
2649 }
2650
2651 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2652   \__stex_terms_maybe_brackets:nn { #3 }{
2653     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2654   }
2655 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 35.)

`_stex_term_math_arg:nnn`

```

2656 \cs_new_protected:Nn \_stex_term_arg:nn {
2657   \stex_unhighlight_term:n {
2658     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2659   }
2660 }

```

```

2661 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2662   \exp_args:Nnx \use:nn
2663     { \int_set:Nn \l__stex_terms_downprec { #2 }
2664       \stex_term_arg:nn { #1 }{ #3 }
2665     }
2666     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2667   }

```

(End definition for `\stex_term_math_arg:nnn`. This function is documented on page 35.)

`\stex_term_math_assoc_arg:nnnn`

```

2668 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2669   \clist_set:Nn \l_tmpa_clist{ #4 }
2670   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2671     \tl_set:Nn \l_tmpa_tl { #4 }
2672   }{
2673     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2674     \clist_reverse:N \l_tmpa_clist
2675     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2676
2677     \clist_map_inline:Nn \l_tmpa_clist {
2678       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2679         \exp_args:Nno
2680           \l_tmpa_cs { ##1 } \l_tmpa_tl
2681       }
2682     }
2683
2684   }
2685   \exp_args:Nnno
2686   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2687 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 35.)

`\stex_term_custom:nn`

```

2688 \cs_new_protected:Nn \stex_term_custom:nn {
2689   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2690   \str_set:Nn \l_tmpa_str { #2 }
2691   \tl_clear:N \l_tmpa_tl
2692   \int_zero:N \l_tmpa_int
2693   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2694   \__stex_terms_custom_loop:
2695 }

```

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

`__stex_terms_custom_loop:`

```

2696 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2697   \bool_set_false:N \l_tmpa_bool
2698   \bool_while_do:nn {
2699     \str_if_eq_p:ee X {
2700       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2701     }
2702   }{
2703     \int_incr:N \l_tmpa_int

```

```

2704 }
2705
2706 \peek_charcode:NTF [ {
2707   % notation/text component
2708   \__stex_terms_custom_component:w
2709 } {
2710   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2711     % all arguments read => finish
2712     \__stex_terms_custom_final:
2713   } {
2714     % arguments missing
2715     \peek_charcode_remove:NTF * {
2716       % invisible, specific argument position or both
2717       \peek_charcode:NTF [ {
2718         % visible specific argument position
2719         \__stex_terms_custom_arg:wn
2720       } {
2721         % invisible
2722         \peek_charcode_remove:NTF * {
2723           % invisible specific argument position
2724           \__stex_terms_custom_arg_inv:wn
2725         } {
2726           % invisible next argument
2727           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2728         }
2729       }
2730     } {
2731       % next normal argument
2732       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2733     }
2734   }
2735 }
2736 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2737 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2738   \bool_set_true:N \l_tmpa_bool
2739   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2740 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2741 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2742   \str_set:Nx \l_tmpb_str {
2743     \str_item:Nn \l_tmpa_str { #1 }
2744   }
2745   \str_case:VnTF \l_tmpb_str {
2746     { X } {
2747       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2748     }
2749     { i } { \__stex_terms_custom_set_X:n { #1 } }
2750     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2751 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2752 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2753 }{}{
2754 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2755 }
2756
2757 \bool_if:nTF \l_tmpa_bool {
2758 \tl_put_right:Nx \l_tmpa_tl {
2759 \stex_annotate_invisible:n {
2760 \stex_term_arg:nn { \int_eval:n { #1 } }
2761 \exp_not:n { { #2 } }
2762 }
2763 }
2764 } {
2765 \tl_put_right:Nx \l_tmpa_tl {
2766 \stex_term_arg:nn { \int_eval:n { #1 } }
2767 \exp_not:n { { #2 } }
2768 }
2769 }
2770
2771 \_stex_terms_custom_loop:
2772 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2773 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2774 \str_set:Nx \l_tmpa_str {
2775 \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2776 X
2777 \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2778 }
2779 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2780 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2781 \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2782 \_stex_terms_custom_loop:
2783 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2784 \cs_new_protected:Nn \_stex_terms_custom_final: {
2785 \int_compare:nNnTF \l_tmpb_int = 0 {
2786 \exp_args:Nnno \stex_term_oms:nnn
2787 }{
2788 \str_if_in:NnTF \l_tmpa_str {b} {
2789 \exp_args:Nnno \stex_term_ombind:nnn
2790 } {
2791 \exp_args:Nnno \stex_term_oma:nnn
2792 }
2793 }

```

```

2794 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2795 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

2796 \NewDocumentCommand \symref { m m }{
2797   \let\compemph_uri_prev:\compemph@uri
2798   \let\compemph@uri\symrefemph@uri
2799   \STEXsymbol{#1}! [#2]
2800   \let\compemph@uri\compemph_uri_prev:
2801 }
2802
2803 \keys_define:nn { stex / symname } {
2804   post      .str_set_x:N    = \l_stex_symname_post_str
2805 }
2806
2807 \cs_new_protected:Nn \stex_symname_args:n {
2808   \str_clear:N \l_stex_symname_post_str
2809   \keys_set:nn { stex / symname } { #1 }
2810 }
2811
2812 \NewDocumentCommand \symname { 0{} m }{
2813   \stex_symname_args:n { #1 }
2814   \stex_get_symbol:n { #2 }
2815   \str_set:Nx \l_tmpa_str {
2816     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2817   }
2818   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2819
2820   \let\compemph_uri_prev:\compemph@uri
2821   \let\compemph@uri\symrefemph@uri
2822   \exp_args:NNx \use:nn
2823   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2824     \l_tmpa_str \l_stex_symname_post_str
2825   ] }
2826   \let\compemph@uri\compemph_uri_prev:
2827 }

```

(End definition for \symref and \symname. These functions are documented on page 35.)

30.3 Notation Components

```

2828 <@@=stex_notationcomps>

```

\stex_highlight_term:nn

```

2829
2830 \str_new:N \l_stex_current_symbol_str
2831 \cs_new_protected:Nn \stex_highlight_term:nn {
2832   \exp_args:Nnx
2833   \use:nn {
2834     \str_set:Nx \l_stex_current_symbol_str { #1 }
2835     #2
2836   } {

```

```

2837 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2838 { \l_stex_current_symbol_str }
2839 }
2840 }
2841
2842 \cs_new_protected:Nn \stex_unhighlight_term:n {
2843 % \latexml_if:TF {
2844 % #1
2845 % } {
2846 % \rustex_if:TF {
2847 % #1
2848 % } {
2849 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2850 % }
2851 % }
2852 }

```

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

```

\comp
\compemph@uri 2853 \cs_new_protected:Npn \comp #1 {
\compemph 2854 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 2855 \rustex_if:TF {
\defemph@uri 2856 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 2857 }{
\symrefemph@uri 2858 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2859 }
2860 }
2861 }
2862
2863 \cs_new_protected:Npn \compemph@uri #1 #2 {
2864 \compemph{ #1 }
2865 }
2866
2867
2868 \cs_new_protected:Npn \compemph #1 {
2869 \textcolor{blue}{#1}
2870 }
2871
2872 \cs_new_protected:Npn \defemph@uri #1 #2 {
2873 \defemph{#1}
2874 }
2875
2876 \cs_new_protected:Npn \defemph #1 {
2877 \textbf{#1}
2878 }
2879
2880 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2881 \symrefemph{#1}
2882 }
2883
2884 \cs_new_protected:Npn \symrefemph #1 {
2885 \textbf{#1}
2886 }

```

(End definition for `\comp` and others. These functions are documented on page 37.)

`\ellipses`

```
2887 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2888 \bool_new:N \l_stex_inarray_bool
\parrayline 2889 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2890 \NewDocumentCommand \parray { m m } {
\parraycell 2891 \begin{group}
2892 \bool_set_true:N \l_stex_inarray_bool
2893 \begin{array}{#1}
2894 #2
2895 \end{array}
2896 \end{group}
2897 }
2898
2899 \NewDocumentCommand \prmatrix { m } {
2900 \begin{group}
2901 \bool_set_true:N \l_stex_inarray_bool
2902 \begin{matrix}
2903 #1
2904 \end{matrix}
2905 \end{group}
2906 }
2907
2908 \def \maybepline {
2909 \bool_if:NT \l_stex_inarray_bool {\hline}
2910 }
2911
2912 \def \parrayline #1 #2 {
2913 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2914 }
2915
2916 \def \pmrow #1 { \parrayline{}{ #1 } }
2917
2918 \def \parraylineh #1 #2 {
2919 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2920 }
2921
2922 \def \parraycell #1 {
2923 #1 \bool_if:NT \l_stex_inarray_bool {&}
2924 }

```

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

```
2925 \end{package}
```

Chapter 31

STEX -Structural Features Implementation

```
2926 <*package>
2927
2928 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2929
2930 <@@=stex_features>
      Warnings and error messages
2931
```

31.1 Imports with modification

```
2932 \seq_new:N \l_stex_implicit_morphisms_seq
2933 \NewDocumentCommand \implicitmorphism { 0{} m m}{
2934   \stex_import_module_uri:nn { #1 } { #2 }
2935   \stex_debug:nn{implicits}{
2936     Implicit~morphism:~
2937     \l_stex_module_ns_str ? \l__stex_features_name_str
2938   }
2939   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
2940     \l_stex_module_ns_str ? \l__stex_features_name_str
2941   }{
2942     \msg_error:nnn{stex}{error/conflictingmodules}{
2943       \l_stex_module_ns_str ? \l__stex_features_name_str
2944     }
2945   }
2946
2947   % TODO
2948
2949
2950
2951   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
2952     \l_stex_module_ns_str ? \l__stex_features_name_str
2953   }
```



```

2954 }
2955

```

31.2 The feature environment

structural@feature

```

2956
2957 \NewDocumentEnvironment{structural@feature}{ m m m }{
2958   \stex_if_in_module:F {
2959     \msg_set:nnn{stex}{error/nomodule}{
2960       Structural~Feature~has~to~occur~in~a~module:\\
2961       Feature~#2~of~type~#1\\
2962       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2963     }
2964     \msg_error:nn{stex}{error/nomodule}
2965   }
2966
2967   \str_set:Nx \l_stex_module_name_str {
2968     \prop_item:Nn \l_stex_current_module_prop
2969       { name } / #2 - feature
2970   }
2971
2972   \str_set:Nx \l_stex_module_ns_str {
2973     \prop_item:Nn \l_stex_current_module_prop
2974       { ns }
2975   }
2976
2977
2978   \str_clear:N \l_tmpa_str
2979   \seq_clear:N \l_tmpa_seq
2980   \tl_clear:N \l_tmpa_tl
2981   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2982     origname = #2,
2983     name      = \l_stex_module_name_str ,
2984     ns        = \l_stex_module_ns_str ,
2985     imports   = \exp_not:o { \l_tmpa_seq } ,
2986     constants = \exp_not:o { \l_tmpa_seq } ,
2987     content   = \exp_not:o { \l_tmpa_tl } ,
2988     file      = \exp_not:o { \g_stex_currentfile_seq } ,
2989     lang      = \l_stex_module_lang_str ,
2990     sig       = \l_tmpa_str ,
2991     meta      = \l_tmpa_str ,
2992     feature   = #1 ,
2993   }
2994
2995   \stex_if_smsmode:TF {
2996     \stex_smsmode_set_codes:
2997   } {
2998     \begin{stex_annotate_env}{ feature:#1 }{}
2999     \stex_annotate_invisible:nnn{header}{}{ #3 }
3000   }
3001 }{
3002   \str_set:Nx \l_tmpa_str {
3003     c_stex_feature_

```

```

3004     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3005     \prop_item:Nn \l_stex_current_module_prop { name }
3006     _prop
3007 }
3008 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3009 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3010 \stex_if_smsmode:TF {
3011     \exp_args:Nx \stex_add_to_sms:n {
3012         \prop_gset_from_keyval:cn {
3013             c_stex_feature_
3014             \prop_item:Nn \l_stex_current_module_prop { ns } ?
3015             \prop_item:Nn \l_stex_current_module_prop { name }
3016             _prop
3017         } {
3018             origname = #2,
3019             name     = \prop_item:cn { \l_tmpa_str } { name } ,
3020             ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
3021             imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3022             constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3023             content  = \prop_item:cn { \l_tmpa_str } { content } ,
3024             file     = \prop_item:cn { \l_tmpa_str } { file } ,
3025             lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3026             sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3027             meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3028             feature  = \prop_item:cn { \l_tmpa_str } { feature }
3029         }
3030     }
3031 } {
3032     \end{stex_annotate_env}
3033 }
3034 }
3035

```

31.3 Features

structure

```

3036
3037 \prop_new:N \l_stex_all_structures_prop
3038
3039 \keys_define:nn { stex / features / structure } {
3040     name .str_set_x:N = \l__stex_features_structure_name_str ,
3041 }
3042
3043 \cs_new_protected:Nn \__stex_features_structure_args:n {
3044     \str_clear:N \l__stex_features_structure_name_str
3045     \keys_set:nn { stex / features / structure } { #1 }
3046 }
3047
3048 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3049 % \__stex_features_structure_args:n { ##1 }
3050 % \str_if_empty:NT \l__stex_features_structure_name_str {
3051 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3052 % }

```

```

3053 %} {
3054 %
3055 %}
3056
3057 \NewDocumentEnvironment{mathstructure}{0}{m}{
3058   \_stex_features_structure_args:n { #1 }
3059   \str_if_empty:NT \l__stex_features_structure_name_str {
3060     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3061   }
3062   \exp_args:Nnnx
3063   \begin{structural@feature}{ structure }
3064     { \l__stex_features_structure_name_str }{}
3065     \seq_clear:N \l_tmpa_seq
3066     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3067
3068   }{
3069     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3070     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3071     \str_set:Nx \l_tmpa_str {
3072       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3073       \prop_item:Nn \l_stex_current_module_prop { name }
3074     }
3075     \seq_map_inline:Nn \l_tmpa_seq {
3076       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3077     }
3078     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3079     \exp_args:Nnx
3080     \AddToHookNext { env / mathstructure / after }{
3081       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3082         \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3083       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3084       \STEXexport {
3085         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3086           {\prop_item:Nn \l_stex_current_module_prop { origname }}
3087           {\l_tmpa_str}
3088         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3089           {#2}{\l_tmpa_str}
3090         % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3091         %   \prop_item:Nn \l_stex_current_module_prop { origname },
3092         %   \l_tmpa_str
3093         % }
3094         % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3095         %   #2,\l_tmpa_str
3096         % }
3097         % \tl_set:cx { #2 } {
3098         %   \stex_invoke_structure:n { \l_tmpa_str }
3099         % }
3100       }
3101
3102   \end{structural@feature}
3103   % \g_stex_last_feature_prop
3104 }

```

\instantiate

```

3105 \seq_new:N \l__stex_features_structure_field_seq
3106 \str_new:N \l__stex_features_structure_field_str
3107 \str_new:N \l__stex_features_structure_def_tl
3108 \prop_new:N \l__stex_features_structure_prop
3109 \NewDocumentCommand \instantiate { m O{} m }{
3110   \stex_smsmode_set_codes:
3111   \prop_get:NnN \l__stex_all_structures_prop {#1} \l_tmpa_str
3112   \prop_set_eq:Nc \l__stex_features_structure_prop {
3113     c_stex_feature_\l_tmpa_str _prop
3114   }
3115   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3116   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3117     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3118     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3119       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3120       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3121         {!} \l_tmpa_tl
3122       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3123         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3124         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3125         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3126       }{
3127         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3128         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3129         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3130           \l_tmpa_tl
3131         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3132           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3133           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3134         }{
3135           \tl_clear:N \l_tmpb_tl
3136         }
3137       }
3138     }{
3139       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3140       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3141         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3142         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3143         \tl_clear:N \l_tmpa_tl
3144       }{
3145         % TODO throw error
3146       }
3147     }
3148     % \l_tmpa_str: name
3149     % \l_tmpa_tl: definiens
3150     % \l_tmpb_tl: notation
3151     \tl_if_empty:NT \l__stex_features_structure_field_str {
3152       % TODO throw error
3153     }
3154     \str_clear:N \l_tmpb_str
3155
3156     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3157     \seq_map_inline:Nn \l_tmpa_seq {
3158       \seq_set_split:Nnn \l_tmpb_seq ? { #####1 }

```

```

3159     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3160     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3161         \seq_map_break:n {
3162             \str_set:Nn \l_tmpb_str { ####1 }
3163         }
3164     }
3165 }
3166 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3167 \l_tmpb_str
3168
3169 \tl_if_empty:NTF \l_tmpb_tl {
3170     \tl_if_empty:NF \l_tmpa_tl {
3171         \exp_args:Nx \use:n {
3172             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3173         }
3174     }
3175 }{
3176     \tl_if_empty:NTF \l_tmpa_tl {
3177         \exp_args:Nx \use:n {
3178             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3179         }
3180     }
3181 }{
3182     \exp_args:Nx \use:n {
3183         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3184         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3185     }
3186 }
3187 }
3188 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3189 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3190 % #3/\l__stex_features_structure_field_str
3191 % \par
3192 % \expandafter\present\csname
3193 %     l_stex_symdecl_
3194 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
3195 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3196 % #3/\l__stex_features_structure_field_str
3197 % _prop
3198 % \endcsname
3199 }
3200
3201 \tl_clear:N \l__stex_features_structure_def_tl
3202
3203 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3204 \seq_map_inline:Nn \l_tmpa_seq {
3205     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3206     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3207     \exp_args:Nx \use:n {
3208         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3209
3210         }
3211     }
3212 }

```

```

3213 \prop_if_exist:cF {
3214   l_stex_symdecl_
3215   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3216   \prop_item:Nn \l_stex_current_module_prop {name} ?
3217   #3/\l_tmpa_str
3218   _prop
3219 }{
3220   \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3221   \l_tmpb_str
3222   \exp_args:Nx \use:n {
3223     \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3224   }
3225 }
3226 }
3227
3228 \symdecl*[type={\STEXsymbol{module-type}}{
3229   \_stex_term_math_oms:nnnn {
3230     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3231     \prop_item:Nn \l__stex_features_structure_prop {name}
3232     }{}{0}{}
3233   }{}{#3}
3234 }
3235 % TODO: -> sms file
3236
3237 \tl_set:cx{ #3 }{
3238   \stex_invoke_structure:nnn {
3239     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3240     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3241   } {
3242     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3243     \prop_item:Nn \l__stex_features_structure_prop {name}
3244   }
3245 }
3246
3247 }

```

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

\stex_invoke_structure:nnn

```

3248 % #1: URI of the instance
3249 % #2: URI of the instantiated module
3250 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3251   \tl_if_empty:nTF{ #3 }{
3252     \prop_set_eq:Nc \l__stex_features_structure_prop {
3253       c_stex_feature_ #2 _prop
3254     }
3255     \tl_clear:N \l_tmpa_tl
3256     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3257     \seq_map_inline:Nn \l_tmpa_seq {
3258       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3259       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3260       \cs_if_exist:cT {
3261         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3262       }{

```

```

3263     \tl_if_empty:NF \l_tmpa_tl {
3264         \tl_put_right:Nn \l_tmpa_tl {,}
3265     }
3266     \tl_put_right:Nx \l_tmpa_tl {
3267         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3268     }
3269 }
3270 }
3271 \exp_args:No \mathstruct \l_tmpa_tl
3272 }{
3273     \stex_invoke_symbol:n{#1/#3}
3274 }
3275 }

```

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

```

3276 </package>

```

Chapter 32

STEX -Statements Implementation

```
3277 <*package>
3278
3279 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3280
3281 \protected\def\ignorespacesandpars{
3282   \begingroup\catcode13=10\relax
3283   \@ifnextchar\par{
3284     \endgroup\expandafter\ignorespacesandpars\@gobble
3285   }{
3286     \endgroup
3287   }
3288 }
3289
3290 <@@=stex_statements>
3291
3292   Warnings and error messages
```

\titleemph

```
3292 \def\titleemph#1{\textbf{#1}}
```

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

32.1 Definitions

definiendum

```
3293 \keys_define:nn {stex / definiendum }{
3294   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3295   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3296   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3297 }
3298 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3299   \str_clear:N \l__stex_statements_definiendum_root_str
3300   \tl_clear:N \l__stex_statements_definiendum_post_tl
3301   \str_clear:N \l__stex_statements_definiendum_gfa_str
```



```

3302 \keys_set:nn { stex / definiendum } { #1 }
3303 }
3304 \NewDocumentCommand \definiendum { 0{ } m m } {
3305   \__stex_statements_definiendum_args:n { #1 }
3306   \stex_get_symbol:n { #2 }
3307   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3308   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3309     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3310       \tl_set:Nn \l_tmpa_tl { #3 }
3311     } {
3312       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3313       \tl_set:Nn \l_tmpa_tl {
3314         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3315       }
3316     }
3317   } {
3318     \tl_set:Nn \l_tmpa_tl { #3 }
3319   }
3320
3321   % TODO root
3322   \rustex_if:TF {
3323     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3324   } {
3325     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3326   }
3327 }
3328 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3329 \NewDocumentCommand \definame { 0{ } m } {
3330   \__stex_statements_definiendum_args:n { #1 }
3331   % TODO: root
3332   \stex_get_symbol:n { #2 }
3333   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3334   \str_set:Nx \l_tmpa_str {
3335     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3336   }
3337   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3338   \rustex_if:TF {
3339     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3340       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3341     }
3342   } {
3343     \defemph@uri {
3344       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3345     } { \l_stex_get_symbol_uri_str }
3346   }
3347 }
3348 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3349
3350 \keys_define:nn {stex / sdefinition }{
3351   type      .str_set_x:N = \sdefinitiontype,
3352   id        .str_set_x:N = \sdefinitionid,
3353   title     .tl_set:N     = \sdefinitiontitle
3354 }
3355 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3356   \str_clear:N \sdefinitiontype
3357   \str_clear:N \sdefinitionid
3358   \tl_clear:N \sdefinitiontitle
3359   \keys_set:nn { stex / sdefinition }{ #1 }
3360 }
3361
3362 \NewDocumentEnvironment{sdefinition}{0{}}{
3363   \__stex_statements_sdefinition_args:n{ #1 }
3364   \stex_reactivate_macro:N \definiendum
3365   \stex_reactivate_macro:N \definame
3366   \stex_smsmode_set_codes:
3367   \clist_set:No \l_tmpa_clist \sdefinitiontype
3368   \tl_clear:N \l_tmpa_tl
3369   \clist_map_inline:Nn \l_tmpa_clist {
3370     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3371       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3372     }
3373   }
3374   \tl_if_empty:NTF \l_tmpa_tl {
3375     \__stex_statements_sdefinition_start:
3376   }{
3377     \l_tmpa_tl
3378   }
3379   \stex_ref_new_doc_target:n \sdefinitionid
3380   \stex_if_smsmode:F {
3381     \exp_args:Nnnx
3382     \begin{stex_annotate_env}{definition}{}
3383     \str_if_empty:NF \sdefinitiontype {
3384       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3385     }
3386   }
3387 }{
3388   \stex_if_smsmode:F {
3389     \end{stex_annotate_env}
3390   }
3391   \clist_set:No \l_tmpa_clist \sdefinitiontype
3392   \tl_clear:N \l_tmpa_tl
3393   \clist_map_inline:Nn \l_tmpa_clist {
3394     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3395       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3396     }
3397   }
3398   \tl_if_empty:NTF \l_tmpa_tl {
3399     \__stex_statements_sdefinition_end:
3400   }{
3401     \l_tmpa_tl

```

```

3402 }
3403 }

```

`\stexpatchdefinition`

```

3404 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3405   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3406     ~(\sdefinitiontitle)
3407   }~}
3408 }
3409 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3410
3411 \newcommand\stexpatchdefinition[3] [] {
3412   \str_set:Nx \l_tmpa_str{ #1 }
3413   \str_if_empty:NTF \l_tmpa_str {
3414     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3415     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3416   }{
3417     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3418     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3419   }
3420 }

```

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

`\inlinedef inline:`

```

3421 \NewDocumentCommand \inlinedef { m } {
3422   \begingroup
3423   \stex_reactivate_macro:N \definiendum
3424   \stex_reactivate_macro:N \definame
3425   \stex_ref_new_doc_target:n{
3426     #1
3427   }
3428 }

```

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

32.2 Assertions

`sassertion`

```

3429
3430 \keys_define:nn {stex / sassertion }{
3431   type      .str_set_x:N = \sassertiontype,
3432   id        .str_set_x:N = \sassertionid,
3433   title     .tl_set:N     = \sassertiontitle ,
3434   name      .str_set_x:N = \sassertionname
3435 }
3436 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3437   \str_clear:N \sassertiontype
3438   \str_clear:N \sassertionid
3439   \str_clear:N \sassertionname
3440   \tl_clear:N \sassertiontitle
3441   \keys_set:nn { stex / sassertion }{ #1 }
3442 }

```

```

3443
3444 \tl_new:N \g__stex_statements_aftergroup_tl
3445
3446 \NewDocumentEnvironment{sassertion}{0{}}{
3447   \__stex_statements_sassertion_args:n{ #1 }
3448   \stex_smsmode_set_codes:
3449   \clist_set:No \l_tmpa_clist \sassertiontype
3450   \tl_clear:N \l_tmpa_tl
3451   \clist_map_inline:Nn \l_tmpa_clist {
3452     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3453       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3454     }
3455   }
3456   \tl_if_empty:NTF \l_tmpa_tl {
3457     \__stex_statements_sassertion_start:
3458   }{
3459     \l_tmpa_tl
3460   }
3461   \stex_ref_new_doc_target:n \sassertionid
3462   \stex_if_smsmode:F {
3463     \exp_args:Nnnx
3464     \begin{stex_annotate_env}{assertion}{}
3465     \str_if_empty:NF \sassertiontype {
3466       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3467     }
3468   }
3469   }{
3470     \stex_if_smsmode:F {
3471       \end{stex_annotate_env}
3472     }
3473     \clist_set:No \l_tmpa_clist \sassertiontype
3474     \tl_clear:N \l_tmpa_tl
3475     \clist_map_inline:Nn \l_tmpa_clist {
3476       \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3477         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3478       }
3479     }
3480     \tl_if_empty:NTF \l_tmpa_tl {
3481       \__stex_statements_sassertion_end:
3482     }{
3483       \l_tmpa_tl
3484     }
3485     \str_if_empty:NF \sassertionname {
3486       \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3487         \symdecl*{\sassertionname}
3488       }
3489       \aftergroup\g__stex_statements_aftergroup_tl
3490     }
3491   }

```

\stexpatchassertion

```

3492
3493 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3494   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3495     (\sassertiontitle)
3496   }~}
3497 }
3498 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3499
3500 \newcommand\stexpatchassertion[3] [] {
3501   \str_set:Nx \l_tmpa_str{ #1 }
3502   \str_if_empty:NTF \l_tmpa_str {
3503     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3504     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3505   }{
3506     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3507     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3508   }
3509 }

```

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

\inlineass inline:

```

3510 \NewDocumentCommand \inlineass { m } {
3511   \begingroup
3512   \stex_ref_new_doc_target:n{
3513     #1
3514   \endgroup
3515 }

```

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

32.3 Examples

sexample

```

3516
3517 \keys_define:nn {stex / sexample }{
3518   type      .str_set_x:N = \exampletype,
3519   id        .str_set_x:N = \sexampleid,
3520   title     .tl_set:N = \sexampletile,
3521   for       .clist_set:N = \sexamplefor,
3522 }
3523 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3524   \str_clear:N \sexampletype
3525   \str_clear:N \sexampleid
3526   \tl_clear:N \sexampletile
3527   \clist_clear:N \sexamplefor
3528   \keys_set:nn { stex / sexample }{ #1 }
3529 }
3530
3531 \NewDocumentEnvironment{sexample}{0{}}{
3532   \__stex_statements_sexample_args:n{ #1 }
3533   \stex_smsmode_set_codes:
3534   \clist_set:Nn \l_tmpa_clist \sexampletype
3535   \tl_clear:N \l_tmpa_tl
3536   \clist_map_inline:Nn \l_tmpa_clist {
3537     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3538     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3539   }
3540 }
3541 \tl_if_empty:NTF \l_tmpa_tl {
3542   \__stex_statements_sexample_start:
3543 }{
3544   \l_tmpa_tl
3545 }
3546 \stex_ref_new_doc_target:n \sexampleid
3547 \stex_if_smsmode:F {
3548   \seq_clear:N \l_tmpa_seq
3549   \clist_map_inline:Nn \sexamplefor {
3550     \str_if_eq:nnF{ ##1 }{}{
3551       \stex_get_symbol:n { ##1 }
3552       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3553         \l_stex_get_symbol_uri_str
3554       }
3555     }
3556   }
3557   \exp_args:Nnnx
3558   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3559   \str_if_empty:NF \sexamplotype {
3560     \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
3561   }
3562 }
3563 }{
3564   \stex_if_smsmode:F {
3565     \end{stex_annotate_env}
3566   }
3567   \clist_set:Nn \l_tmpa_clist \sexamplotype
3568   \tl_clear:N \l_tmpa_tl
3569   \clist_map_inline:Nn \l_tmpa_clist {
3570     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3571       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3572     }
3573   }
3574   \tl_if_empty:NTF \l_tmpa_tl {
3575     \__stex_statements_sexample_end:
3576   }{
3577     \l_tmpa_tl
3578   }
3579 }

```

\stexpatchexample

```

3580
3581 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3582   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3583     (\sexamplotype)
3584   }~}
3585 }
3586 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3587
3588 \newcommand\stexpatchexample[3][ ] {
3589   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3590 \str_if_empty:NTF \l_tmpa_str {
3591   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3592   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3593 }{
3594   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3595   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3596 }
3597 }

```

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

`\inlineex` inline:

```

3598 \NewDocumentCommand \inlineex { m } {
3599   \beginngroup
3600   \stex_ref_new_doc_target:n{
3601     #1
3602   \endgroup
3603 }

```

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

32.4 Logical Paragraphs

`sparagraph`

```

3604 \keys_define:nn { stex / spparagraph } {
3605   id      .str_set_x:N = \sparagraphid ,
3606   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3607   type    .str_set_x:N = \sparagraphtype ,
3608   for     .str_set_x:N = \sparagraphfor ,
3609   from    .tl_set_x:N   = \sparagraphfrom ,
3610   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
3611   name    .str_set:N    = \sparagraphname
3612 }
3613
3614 \cs_new_protected:Nn \stex_sparagraph_args:n {
3615   \tl_clear:N \l_stex_sparagraph_title_tl
3616   \tl_clear:N \sparagraphfrom
3617   \tl_clear:N \l_stex_sparagraph_start_tl
3618   \str_clear:N \sparagraphid
3619   \str_clear:N \sparagraphtype
3620   \str_clear:N \sparagraphfor
3621   \str_clear:N \sparagraphname
3622   \keys_set:nn { stex / spparagraph } { #1 }
3623 }
3624 \newif\if@in@omtext\@in@omtextfalse
3625
3626 \NewDocumentEnvironment {sparagraph} { 0{} } {
3627   \stex_sparagraph_args:n { #1 }
3628   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3629     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3630   }{
3631     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3632   }

```

```

3633 \in@omtexttrue
3634 \stex_smsmode_set_codes:
3635 \clist_set:No \l_tmpa_clist \sparagraphtype
3636 \tl_clear:N \l_tmpa_tl
3637 \clist_map_inline:Nn \l_tmpa_clist {
3638   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3639     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3640   }
3641 }
3642 \tl_if_empty:NTF \l_tmpa_tl {
3643   \__stex_statements_sparagraph_start:
3644 }{
3645   \l_tmpa_tl
3646 }
3647 \stex_ref_new_doc_target:n \sparagraphid
3648 \stex_if_smsmode:F {
3649   \exp_args:Nnnx
3650   \begin{stex_annotate_env}{paragraph}{}
3651   \str_if_empty:NF \sparagraphtype {
3652     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
3653   }
3654 }
3655 \ignorespacesandpars
3656 }{
3657   \stex_if_smsmode:F {
3658     \end{stex_annotate_env}
3659   }
3660   \clist_set:No \l_tmpa_clist \sparagraphtype
3661   \tl_clear:N \l_tmpa_tl
3662   \clist_map_inline:Nn \l_tmpa_clist {
3663     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
3664       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
3665     }
3666   }
3667   \tl_if_empty:NTF \l_tmpa_tl {
3668     \__stex_statements_sparagraph_end:
3669   }{
3670     \l_tmpa_tl
3671   }
3672   \str_if_empty:NF \sparagraphname {
3673     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3674       \symdecl*{\sparagraphname}
3675     }
3676     \aftergroup\g__stex_statements_aftergroup_tl
3677   }
3678 }

```

\stexpatchparagraph

```

3679
3680 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
3681   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3682     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
3683       \titleemph{\l_stex_sparagraph_title_tl}:~
3684     }

```



```

3685   }{
3686   \titleemph{\l_stex_sparagraph_start_tl}~
3687   }
3688 }
3689 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
3690
3691 \newcommand\stexpatchparagraph[3] [] {
3692   \str_set:Nx \l_tmpa_str{ #1 }
3693   \str_if_empty:NTF \l_tmpa_str {
3694     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
3695     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
3696   }{
3697     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
3698     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
3699   }
3700 }

```

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

symboldoc

```

3701 \NewDocumentEnvironment{symboldoc}{ m }{
3702   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3703   \seq_clear:N \l_tmpb_seq
3704   \seq_map_inline:Nn \l_tmpa_seq {
3705     \str_if_eq:nnF{ ##1 }{}{
3706       \stex_get_symbol:n { ##1 }
3707       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3708         \l_stex_get_symbol_uri_str
3709       }
3710     }
3711   }
3712   \par
3713   \exp_args:Nnnx
3714   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3715 }{
3716   \end{stex_annotate_env}
3717 }
3718 \</package>

```

Chapter 33

The Implementation

33.1 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

```
3719 <*package>
3720 <@@=stex_sproof>
3721
3722 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3723
```

33.2 Proofs

We first define some keys for the proof environment.

```
3724 \keys_define:nn { stex / spf } {
3725   id          .str_set:N = \l__stex_sproof_spf_id_str,
3726   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3727   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3728   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3729   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3730   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3731   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3732   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3733   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3734   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3735 }
3736 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3737   \str_clear:N \l__stex_sproof_spf_id_str
3738   \tl_clear:N \l__stex_sproof_spf_display_tl
3739   \tl_clear:N \l__stex_sproof_spf_for_tl
3740   \tl_clear:N \l__stex_sproof_spf_from_tl
3741   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3742   \tl_clear:N \l__stex_sproof_spf_type_tl
3743   \tl_clear:N \l__stex_sproof_spf_title_tl

```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

3744 \tl_clear:N \l__stex_sproof_spf_continues_tl
3745 \tl_clear:N \l__stex_sproof_spf_functions_tl
3746 \tl_clear:N \l__stex_sproof_spf_method_tl
3747 \keys_set:nn { stex / spf }{ #1 }
3748 }

```

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

```

3749 \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.

```

3750 \newcount\count_ten
3751 \newenvironment{pst@with@label}[1]{
3752   \edef\pst@label{#1}
3753   \advance\count_ten by 1\relax
3754   \count_ten=1
3755 }{
3756   \advance\count_ten by -1\relax
3757 }

```

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

```

3758 \def\the@pst@label{
3759   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3760 }

```

(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.

```

3761 \keys_define:nn { stex / pstlabel }{
3762   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3763   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3764   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3765 }
3766 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

3767 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3768 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3769 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3770 }
3771 \__stex_sproof_pstlabel_args:n {}
3772 \newcommand\setpstlabelstyle[1]{
3773   \__stex_sproof_pstlabel_args:n {#1}
3774 }
3775 \newcommand\setpstlabelstyledefault{%
3776   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3777 }

```

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

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

```

3778 \ExplSyntaxOff
3779 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3780 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3781 \def\pst@make@label@short#1#2{#2}
3782 \def\pst@make@label@empty#1#2{}
3783 \ExplSyntaxOn
3784 \def\pstlabelstyle#1{%
3785   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3786 }%
3787 \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.

```

3788 \def\next@pst@label{%
3789   \global\advance\count\count10 by 1%
3790 }%

```

(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

```

3791 \def\sproof@box{
3792   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3793 }
3794 \def\spf@proofend{\sproof@box}
3795 \def\sproofend{
3796   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3797     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3798   }
3799 }
3800 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3801 \def\spf@proofsketch@kw{Proof Sketch}
3802 \def\spf@proof@kw{Proof}
3803 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3804 \cs_if_exist:NT \bbl@loaded {
3805   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3806   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3807     \input{proof-ngerman.lda}
3808   }
3809   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3810     \input{proof-finnish.lda}
3811   }
3812   \clist_if_in:NnT \l_tmpa_clist {french}{
3813     \input{proof-french.lda}
3814   }
3815   \clist_if_in:NnT \l_tmpa_clist {russian}{
3816     \input{proof-russian.lda}
3817   }
3818 }
3819

```

spfsketch

```

3820 \newcommand\spfsketch[2] [] {
3821   \__stex_sproof_spf_args:n{#1}
3822   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3823     \titleemph{
3824       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3825         \spf@proofsketch@kw
3826       }{
3827         \l__stex_sproof_spf_type_tl
3828       }
3829     }:
3830   }
3831   {-#2}
3832   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3833   \sproofend
3834 }

```

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

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

```

3835 \newenvironment{spfeq}[2] [] {
3836   \__stex_sproof_spf_args:n{#1}
3837   %\sref@target
3838   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3839     \titleemph{
3840       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3841         \spf@proof@kw
3842       }{
3843         \l__stex_sproof_spf_type_tl
3844       }
3845     }:

```

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

¹⁵EDNOTE: document above

```

3846 }
3847 {~#2}
3848 \begin{displaymath}\begin{array}{rcll}
3849 }{
3850 \end{array}\end{displaymath}
3851 }

```

(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.

```

3852 \newenvironment{spf@proof}[2][]{
3853   \__stex_sproof_spf_args:n{#1}
3854   %\sref@target
3855   \count_ten=10
3856   \par\noindent
3857   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3858     \titleemph{
3859       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3860         \spf@proof@kw
3861       }{
3862         \l__stex_sproof_spf_type_tl
3863       }
3864     }:
3865   }
3866   {~#2}
3867   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3868   \def\pst@label{}
3869   \newcount\pst@count% initialize the labeling mechanism
3870   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3871   }{
3872     \end{pst@with@label}\end{description}
3873   }
3874   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3875   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3876 \newcommand\spfidea[2][]{
3877   \__stex_sproof_spf_args:n{#1}
3878   \titleemph{
3879     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3880       \l__stex_sproof_spf_type_tl
3881     }:
3882   }~#2
3883   \sproofend
3884 }

```

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

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

spfstep 16

```

3885 \newenvironment{spfstep}[1][]{
3886   \_stex_sproof_spf_args:n{#1}
3887   \@in@omtexttrue
3888   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3889     \item[\the@pst@label]
3890   }
3891   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3892     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3893   }
3894   %\sref@label@id{\pst@label}
3895   \ignorespacesandpars
3896 }{
3897   \next@pst@label\ignorespacesandpars
3898 }

```

sproofcomment

```

3899 \newenvironment{sproofcomment}[1][]{
3900   \_stex_sproof_spf_args:n{#1}
3901   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3902     \item[\the@pst@label]
3903   }
3904 }{
3905   \next@pst@label
3906 }

```

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.

```

3907 \newenvironment{subproof}[2][]{
3908   \_stex_sproof_spf_args:n{#1}
3909   \def\@test{#2}
3910   \ifx\@test\empty\else
3911     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3912       \item[\the@pst@label]
3913     }{#2}
3914   \fi
3915   \begin{pst@with@label}{\pst@label,\number\count_ten}
3916 }{
3917   \end{pst@with@label}\next@pst@label
3918 }

```

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

```

3919 \newenvironment{spfcases}[2][]{
3920   \def\@test{#1}
3921   \ifx\@test\empty
3922     \begin{subproof}[method=by-cases]{#2}
3923   \else
3924     \begin{subproof}[#1,method=by-cases]{#2}
3925   \fi
3926 }{

```

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

```

3927 \end{subproof}
3928 }

```

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

```

3929 \newenvironment{spfcase}[2] [] {
3930   \__stex_sproof_spf_args:n{#1}
3931   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3932     \item[\the@pst@label]
3933   }
3934   \def\@test{#2}
3935   \ifx\@test\@empty
3936   \else
3937     {\titleemph{#2}:~}
3938   \fi
3939   \begin{pst@with@label}{\pst@label,\number\count_ten}
3940   }{
3941     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3942       \sproofend
3943     }
3944     \end{pst@with@label}
3945     \next@pst@label
3946   }

```

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

```

3947 \newcommand\spfcasesketch[3] [] {
3948   \__stex_sproof_spf_args:n{#1}
3949   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3950     \item[\the@pst@label]
3951   }
3952   \def\@test{#2}
3953   \ifx\@test\@empty
3954   \else
3955     {\titleemph{#2}:~}
3956   \fi#3
3957   \next@pst@label
3958 }%

```

33.3 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.

```

3959 \keys_define:nn { stex / just }{
3960   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3961   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
3962   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
3963   args        .tl_set:N   = \l__stex_sproof_just_args_tl
3964 }

```

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

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

justification

```
3965 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3966 \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.

```
3967 \newcommand\justarg[2] [] {#2}
```

```
3968 \end{package}
```

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

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

Chapter 34

STEX -Others Implementation

```
3969 <*package>
3970
3971 %%%%%%%%%%% others.dtx %%%%%%%%%%%
3972
3973 <@@=stex_others>
    Warnings and error messages
3974 % None

\MSC Math subject classifier

3975 \NewDocumentCommand \MSC {m} {
3976 % TODO
3977 }

(End definition for \MSC. This function is documented on page 18.)
    Patching tikzinput, if loaded
3978 \@ifpackageloaded{tikzinput}{
3979 \RequirePackage{stex-tikzinput}
3980 }{}
3981 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
3982 <*package>
3983 <@@=stex_modules>
3984
3985 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3986
3987 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3988 \begingroup
3989 \stex_module_setup:nn{
3990   ns=\c_stex_metatheory_ns_str,
3991   meta=NONE
3992 }{Metatheory}
3993 \stex_reactivate_macro:N \symdecl
3994 \stex_reactivate_macro:N \notation
3995 \stex_reactivate_macro:N \symdef
3996 \ExplSyntaxOff
3997 \csname stex_suppress_html:n\endcsname{
3998   % is-a (a:A, a \in A, a is an A, etc.)
3999   \symdecl[args=ai]{isa}
4000   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4001   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4002   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4003
4004   % bind (\forall, \Pi, \lambda etc.)
4005   \symdecl[args=Bi]{bind}
4006   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4007   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4008   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4009
4010   % dummy variable
4011   \symdecl{dummyvar}
4012   \notation[underscore]{dummyvar}{\comp\_}
4013   \notation[dot]{dummyvar}{\comp\cdot}
4014   \notation[dash]{dummyvar}{\comp{\rm --}}
4015
4016   %fromto (function space, Hom-set, implication etc.)
```

```

4017 \symdecl[args=ai]{fromto}
4018 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4019 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4020
4021 % mapto (lambda etc.)
4022 %\symdecl[args=Bi]{mapto}
4023 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4024 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4025 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4026
4027 % function/operator application
4028 \symdecl[args=ia]{apply}
4029 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4030 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4031
4032 % ‘type’ of all collections (sets, classes, types, kinds)
4033 \symdecl{collection}
4034 \notation[U]{collection}{\comp{\mathcal{U}}}
4035 \notation[set]{collection}{\comp{\textsf{Set}}}
4036
4037 % sequences
4038 \symdecl[args=1]{seqtype}
4039 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4040
4041 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4042 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4043
4044 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
4045 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
4046 % ^ superceded by \aseqfromto and \livar/\uivar
4047
4048 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4049 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4050 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
4051
4052 % letin (‘let’, local definitions, variable substitution)
4053 \symdecl[args=bii]{letin}
4054 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4055 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4056 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4057
4058 % structures
4059 \symdecl*[args=1]{module-type}
4060 \notation{module-type}{\mathtt{MOD} #1}
4061 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4062 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4063
4064 }
4065 \ExplSyntaxOn
4066 \stex_add_to_current_module:n{
4067   \let\nappa\apply
4068   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4069   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4070   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4071 \def\uivar{\csname sequence-index\endcsname[ui]}
4072 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4073 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4074 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4075 }
4076 \__stex_modules_end_module:
4077 \endgroup
4078 \</package>

```

Chapter 36

Tikzinput Implementation

```
4079 <*package>
4080
4081 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4082
4083 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4084 \RequirePackage{l3keys2e}
4085
4086 \keys_define:nn { tikzinput } {
4087   image .bool_set:N = \c_tikzinput_image_bool,
4088   image .default:n = false ,
4089   unknown .code:n = {}
4090 }
4091
4092 \ProcessKeysOptions { tikzinput }
4093
4094 \bool_if:NTF \c_tikzinput_image_bool {
4095   \RequirePackage{graphicx}
4096
4097   \providecommand\usetikzlibrary[]{}
4098   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4099 }{
4100   \RequirePackage{tikz}
4101   \RequirePackage{standalone}
4102
4103   \newcommand \tikzinput [2] [] {
4104     \setkeys{Gin}{#1}
4105     \ifx \Gin@ewidth \Gin@exclamation
4106       \ifx \Gin@eheight \Gin@exclamation
4107         \input { #2 }
4108       \else
4109         \resizebox{!}{ \Gin@eheight }{
4110           \input { #2 }
4111         }
4112       \fi
4113     \else
4114       \ifx \Gin@eheight \Gin@exclamation
4115         \resizebox{ \Gin@ewidth }{!}{
4116           \input { #2 }
```

```

4117     }
4118     \else
4119         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4120             \input { #2 }
4121         }
4122     \fi
4123 \fi
4124 }
4125 }
4126
4127 \newcommand \ctikzinput [2] [] {
4128     \begin{center}
4129         \tikzinput [1] {#2}
4130     \end{center}
4131 }
4132
4133 \@ifpackageloaded{stex}{
4134     \RequirePackage{stex-tikzinput}
4135 }{}
4136
4137 </package>
4138 <*stex>
4139 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4140 \RequirePackage{stex}
4141 \RequirePackage{tikzinput}
4142
4143 \newcommand\mhtikzinput [2] [] {%
4144     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4145     \stex_in_repository:nn\Gin@mhrepos{
4146         \tikzinput [1]{\mhpath{##1}{#2}}
4147     }
4148 }
4149 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4150 </stex>

```

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

Chapter 37

document-structure.sty Implementation

37.1 The OMDoc Class

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

```
4151 \*cls)
4152 \@@=document_structure)
4153 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4154 \RequirePackage{l3keys2e,expl-keystr-compat}
```

37.2 Class Options

To initialize the `omdoc` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
4155 \keys_define:nn{ document-structure / pkg }{
4156   class      .str_set_x:N = \c_document_structure_class_str,
4157   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4158   report     .code:n      = {
4159     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4160     \str_set:Nn \c_document_structure_class_str {report}
4161   },
4162   book       .code:n      = {
4163     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4164     \str_set:Nn \c_document_structure_class_str {book}
4165   },
4166   bookpart   .code:n      = {
4167     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4168     \str_set:Nn \c_document_structure_class_str {book}
4169     \str_set:Nn \c_document_structure_topsect_str {chapter}
4170   },
```



```

4171 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4172 unknown     .code:n      = {
4173   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4174 }
4175 }
4176 \ProcessKeysOptions{ document-structure / pkg }
4177 \str_if_empty:NT \c_document_structure_class_str {
4178   \str_set:Nn \c_document_structure_class_str {article}
4179 }
4180 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4181   {\c_document_structure_class_str}
4182

```

37.3 Beefing up the document environment

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

```

4183 \RequirePackage{omdoc}
4184 \bool_if:NF \c_document_structure_minimal_bool {
4185   \RequirePackage{stex-compatibility}

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.¹⁸

```

4186 \keys_define:nn { document-structure / document }{
4187   id .str_set_x:N = \c_document_structure_document_id_str
4188 }
4189 \let\__document_structure_orig_document=\document
4190 \renewcommand{\document}[1][]{
4191   \keys_set:nn{ document-structure / document }{ #1 }
4192   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4193   \__document_structure_orig_document
4194 }

```

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

```

4195 }
4196 \</cls>

```

37.4 Implementation: OMDoc Package

```

4197 \*package>
4198 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4199 \RequirePackage{expl-keystr-compat,13keys2e}

```

37.5 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

```

4200
4201 \keys_define:nn{ document-structure / pkg }{
4202   class      .str_set_x:N = \c_document_structure_class_str,
4203   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4204   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4205 }
4206 \ProcessKeysOptions{ document-structure / pkg }
4207 \str_if_empty:NT \c_document_structure_class_str {
4208   \str_set:Nn \c_document_structure_class_str {article}
4209 }
4210 \str_if_empty:NT \c_document_structure_topsect_str {
4211   \str_set:Nn \c_document_structure_topsect_str {section}
4212 }

```

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

```

4213 \RequirePackage{xspace}
4214 \RequirePackage{comment}
4215 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4216 \@ifpackageloaded{babel}{
4217   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4218   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4219     \input{omdoc-ngerman.ldf}
4220   }
4221 }{}
4222 %\AfterBabelLanguage{ngerman}{\input{omdoc-ngerman.ldf}}

```

`\section@level`

Finally, we set the `\section@level` macro that governs sectioning. The default is two (corresponding to the `article` class), then we set the defaults for the standard classes `book` and `report` and then we take care of the levels passed in via the `topsect` option.

```

4223 \int_new:N \l_document_structure_section_level_int
4224 \str_case:VnF \c_document_structure_topsect_str {
4225   {part}{
4226     \int_set:Nn \l_document_structure_section_level_int {0}
4227   }
4228   {chapter}{
4229     \int_set:Nn \l_document_structure_section_level_int {1}
4230   }
4231 }{
4232   \str_case:VnF \c_document_structure_class_str {
4233     {book}{
4234       \int_set:Nn \l_document_structure_section_level_int {0}
4235     }
4236     {report}{
4237       \int_set:Nn \l_document_structure_section_level_int {0}
4238     }
4239   }{
4240     \int_set:Nn \l_document_structure_section_level_int {2}
4241   }
4242 }

```

37.6 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

```
4243 \def\current@section@level{document}%
4244 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4245 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4246 \cs_new_protected:Npn \skipomgroup {
4247   \ifcase\l_document_structure_section_level_int
4248   \or\stepcounter{part}
4249   \or\stepcounter{chapter}
4250   \or\stepcounter{section}
4251   \or\stepcounter{subsection}
4252   \or\stepcounter{subsubsection}
4253   \or\stepcounter{paragraph}
4254   \or\stepcounter{subparagraph}
4255   \fi
4256 }
```

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

`blindomgroup`

```
4257 \newcommand\at@begin@blindomgroup[1]{%
4258 \newenvironment{blindomgroup}
4259 {
4260   \int_incr:N\l_document_structure_section_level_int
4261   \at@begin@blindomgroup\l_document_structure_section_level_int
4262 }{}}
```

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

```
4263 \newcommand\omgroup@nonum[2]{
4264   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4265   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4266 }
```

(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.

```
4267 \newcommand\omgroup@num[2]{
```

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

```

4268 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4269   \@nameuse{#1}{#2}
4270 }{
4271   \cs_if_exist:NTF\rdfmata@sectioning{
4272     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4273   }{
4274     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4275   }
4276 }
4277 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4278 }

```

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

omgroup

```

4279 \keys_define:nn { document-structure / omgroup }{
4280   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4281   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4282   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4283   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4284   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4285   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4286   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4287   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4288   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4289   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4290 }
4291 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4292   \str_clear:N \l__document_structure_omgroup_id_str
4293   \str_clear:N \l__document_structure_omgroup_date_str
4294   \clist_clear:N \l__document_structure_omgroup_creators_clist
4295   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4296   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4297   \tl_clear:N \l__document_structure_omgroup_type_tl
4298   \tl_clear:N \l__document_structure_omgroup_short_tl
4299   \tl_clear:N \l__document_structure_omgroup_display_tl
4300   \tl_clear:N \l__document_structure_omgroup_intro_tl
4301   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4302   \keys_set:nn { document-structure / omgroup } { #1 }
4303 }

```

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.

```

4304 \newif\if@mainmatter\@mainmattertrue
4305 \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.

```

4306 \keys_define:nn { document-structure / sectioning }{
4307   name .str_set_x:N = \l__document_structure_sect_name_str ,
4308   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4309   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4310   num .bool_set:N = \l__document_structure_sect_num_bool ,
4311 }

```

```

4312 \cs_new_protected:Nn \__document_structure_sect_args:n {
4313   \str_clear:N \l__document_structure_sect_name_str
4314   \str_clear:N \l__document_structure_sect_ref_str
4315   \bool_set_false:N \l__document_structure_sect_clear_bool
4316   \bool_set_false:N \l__document_structure_sect_num_bool
4317   \keys_set:nn { document-structure / sectioning } { #1 }
4318 }
4319 \newcommand\omdoc@sectioning[3][]{
4320   \__document_structure_sect_args:n {#1 }
4321   \let\omdoc@sect@name\l__document_structure_sect_name_str
4322   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4323   \if@mainmatter% numbering not overridden by frontmatter, etc.
4324     \bool_if:NTF \l__document_structure_sect_num_bool {
4325       \omgroup@num{#2}{#3}
4326     }{
4327       \omgroup@nonum{#2}{#3}
4328     }
4329     \def\current@section@level{\omdoc@sect@name}
4330   \else
4331     \omgroup@nonum{#2}{#3}
4332   \fi
4333 }% 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.

```

4334 \newcommand\omgroup@redefine@addtocontents[1]{%
4335   %\edef\__document_structureimport{#1}%
4336   %\@for\@I:=\__document_structureimport\do{%
4337     %\edef\@path{\csname module@\@I @path\endcsname}%
4338     %\@ifundefined{tf@toc}\relax%
4339     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4340   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4341   %\def\addcontentsline##1##2##3{%
4342     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4343   %\else% hyperref.sty not loaded
4344   %\def\addcontentsline##1##2##3{%
4345     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4346   %\fi
4347 }% 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.

```

4348 \int_new:N \l_document_structure_omgroup_level_int
4349 \newenvironment{omgroup}[2][]{% keys, title
4350 {
4351   \__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.

```

4352 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4353   \omgroup@redefine@addtocontents{
4354     %\@ifundefined{module@id}\used@modules%
4355     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4356     }
4357 }

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

4358 \int_incr:N \l_document_structure_omgroup_level_int
4359 \int_incr:N \l_document_structure_section_level_int
4360 \ifcase\l_document_structure_section_level_int
4361   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4362   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4363   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4364   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4365   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4366   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4367   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4368 \fi
4369 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4370 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4371 }% for customization
4372 {}

```

and finally, we localize the sections

```

4373 \newcommand\omdoc@part@kw{Part}
4374 \newcommand\omdoc@chapter@kw{Chapter}
4375 \newcommand\omdoc@section@kw{Section}
4376 \newcommand\omdoc@subsection@kw{Subsection}
4377 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4378 \newcommand\omdoc@paragraph@kw{paragraph}
4379 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.7 Front and Backmatter

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

`\printindex`

```

4380 \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).

```

4381 \cs_if_exist:NTF\frontmatter{
4382   \let\__document_structure_orig_frontmatter\frontmatter
4383   \let\frontmatter\relax
4384 }{
4385   \tl_set:Nn\__document_structure_orig_frontmatter{
4386     \clearpage
4387     \@mainmatterfalse
4388     \pagenumbering{roman}
4389   }
4390 }
4391 \cs_if_exist:NTF\backmatter{

```

```

4392 \let\__document_structure_orig_backmatter\backmatter
4393 \let\backmatter\relax
4394 }{
4395 \tl_set:Nn\__document_structure_orig_backmatter{
4396 \clearpage
4397 \@mainmatterfalse
4398 \pagenumbering{roman}
4399 }
4400 }

```

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.

```

4401 \newenvironment{frontmatter}{
4402 \__document_structure_orig_frontmatter
4403 }{
4404 \cs_if_exist:NTF\mainmatter{
4405 \mainmatter
4406 }{
4407 \clearpage
4408 \@mainmattertrue
4409 \pagenumbering{arabic}
4410 }
4411 }

```

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

```

4412 \newenvironment{backmatter}{
4413 \__document_structure_orig_backmatter
4414 }{
4415 \cs_if_exist:NTF\mainmatter{
4416 \mainmatter
4417 }{
4418 \clearpage
4419 \@mainmattertrue
4420 \pagenumbering{arabic}
4421 }
4422 }

```

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

```

4423 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4424 \def \c__document_structure_document_str{document}
4425 \newcommand\afterprematurestop{}
4426 \def\prematurestop@endomgroup{
4427 \unless\ifx\@currenvir\c__document_structure_document_str
4428 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4429 \expandafter\prematurestop@endomgroup
4430 \fi
4431 }
4432 \providecommand\prematurestop{

```

```

4433 \message{Stopping~sTeX~processing~prematurely}
4434 \prematurestop@endomgroup
4435 \afterprematurestop
4436 \end{document}
4437 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

4438 \RequirePackage{etoolbox}
4439 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4440 \newrobustcmd\useSGvar[1]{%
4441 \@ifundefined{sTeX@Gvar@#1}
4442 {\PackageError{omdoc}
4443 {The sTeX Global variable #1 is undefined}
4444 {set it with \protect\setSGvar}}
4445 \@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.

```

4446 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4447 \@ifundefined{sTeX@Gvar@#1}
4448 {\PackageError{omdoc}
4449 {The sTeX Global variable #1 is undefined}
4450 {set it with \protect\setSGvar}}
4451 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 38

MiKoSlides – Implementation

38.1 Class and Package Options

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

```
4452 \*cls)
4453 \@@=mikoslides)
4454 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4455 \RequirePackage{l3keys2e,expl-keystr-compatible}
4456
4457 \keys_define:nn{mikoslides / cls}{
4458   class .code:n = {
4459     \PassOptionsToClass{\CurrentOption}{omdoc}
4460     \str_if_eq:nnT{#1}{book}{
4461       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4462     }
4463     \str_if_eq:nnT{#1}{report}{
4464       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4465     }
4466   },
4467   notes .bool_set:N = \c__mikoslides_notes_bool ,
4468   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4469   unknown .code:n = {
4470     \PassOptionsToClass{\CurrentOption}{omdoc}
4471     \PassOptionsToClass{\CurrentOption}{beamer}
4472     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4473   }
4474 }
4475 \ProcessKeysOptions{ mikoslides / cls }
4476 \bool_if:NTF \c__mikoslides_notes_bool {
4477   \PassOptionsToPackage{notes=true}{mikoslides}
4478 }{
4479   \PassOptionsToPackage{notes=false}{mikoslides}
4480 }
4481 \</cls)
```

now we do the same for the mikoslides package.

```

4482 <*package>
4483 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4484 \RequirePackage{l3keys2e,expl-keystr-compat}
4485
4486 \keys_define:nn{mikoslides / pkg}{
4487   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4488   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4489   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4490   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4491   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4492   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4493   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4494   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4495   unknown      .code:n      = {
4496     \PassOptionsToClass{\CurrentOption}{stex}
4497     \PassOptionsToClass{\CurrentOption}{tikzinput}
4498   }
4499 }
4500 \ProcessKeysOptions{ mikoslides / pkg }
4501 \newif\ifnotes
4502 \bool_if:NTF \c__mikoslides_notes_bool {
4503   \notesttrue
4504 }{
4505   \notesfalse
4506 }
4507

```

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

```

4508 \str_if_empty:NTF \c__mikoslides_topsect_str {
4509   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4510 }{
4511   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4512 }
4513 </package>

```

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

```

4514 <*cls>
4515 \bool_if:NTF \c__mikoslides_notes_bool {
4516   \LoadClass{omdoc}
4517 }{
4518   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4519   \newcounter{Item}
4520   \newcounter{paragraph}
4521   \newcounter{subparagraph}
4522   \newcounter{Hfootnote}
4523   \RequirePackage{omdoc}
4524 }

```

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

```

4525 \RequirePackage{mikoslides}
4526 </cls>

```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the `STEX` packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the `STEX`-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```

4527 \*package>
4528 \bool_if:NT \c__mikoslides_notes_bool {
4529   \RequirePackage{a4wide}
4530   \RequirePackage{marginnote}
4531   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4532   \RequirePackage{mdframed}
4533   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4534   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4535 }
4536 \RequirePackage{stex-compatibility}
4537 \RequirePackage{stex-tikzinput}
4538 \RequirePackage{etoolbox}
4539 \RequirePackage{amssymb}
4540 \RequirePackage{amsmath}
4541 \RequirePackage{comment}
4542 \RequirePackage{textcomp}
4543 \RequirePackage{url}
4544 \RequirePackage{graphicx}
4545 \RequirePackage{pgf}

```

38.2 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

```

4546 \bool_if:NT \c__mikoslides_notes_bool {
4547   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4548 }

```

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

```

4549 \newcounter{slide}
4550 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4551 \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.

```

4552 \bool_if:NTF \c__mikoslides_notes_bool {
4553   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
4554 }{
4555   \excludecomment{note}
4556 }

```

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.

```
4557 \bool_if:NT \c__mikoslides_notes_bool {
4558   \newlength{\slideframewidth}
4559   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4560 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4561   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4562     \bool_set_true:N #1
4563   }{
4564     \bool_set_false:N #1
4565   }
4566 }
4567 \keys_define:nn{mikoslides / frame}{
4568   label .str_set_x:N = \l__mikoslides_frame_label_str,
4569   allowframebreaks .code:n = {
4570     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4571   },
4572   allowdisplaybreaks .code:n = {
4573     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4574   },
4575   fragile .code:n = {
4576     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4577   },
4578   shrink .code:n = {
4579     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4580   },
4581   squeeze .code:n = {
4582     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4583   },
4584   t .code:n = {
4585     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4586   },
4587 }
4588 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4589   \str_clear:N \l__mikoslides_frame_label_str
4590   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4591   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4592   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4593   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4594   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4595   \bool_set_true:N \l__mikoslides_frame_t_bool
4596   \keys_set:nn { mikoslides / frame }{ #1 }
4597 }
```

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

```
4598 \renewenvironment{frame}[1][]{
4599   \__mikoslides_frame_args:n{#1}
4600   \sffamily
4601   \stepcounter{slide}
4602   \def\@currentlabel{\theslide}
4603   \str_if_empty:NF \l__mikoslides_frame_label_str {
4604     \label{\l__mikoslides_frame_label_str}
```

```
4605 }
```

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

```
4606 \def\itemize@level{outer}
4607 \def\itemize@outer{outer}
4608 \def\itemize@inner{inner}
4609 \renewcommand\newpage{\addtocounter{framenum}{1}}
4610 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4611 \renewenvironment{itemize}{
4612   \ifx\itemize@level\itemize@outer
4613     \def\itemize@label{$\rhd$}
4614   \fi
4615   \ifx\itemize@level\itemize@inner
4616     \def\itemize@label{$\scriptstyle\rhd$}
4617   \fi
4618   \begin{list}
4619     {\itemize@label}
4620     {\setlength{\labelsep}{.3em}
4621      \setlength{\labelwidth}{.5em}
4622      \setlength{\leftmargin}{1.5em}
4623     }
4624   \edef\itemize@level{\itemize@inner}
4625 }{
4626   \end{list}
4627 }
```

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

```
4628 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4629 }{
4630   \medskip\miko@slidelabel\end{mdframed}
4631 }
```

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

`\frametitle`

```
4632 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4633 }
```

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

EdN:21

`\pause` 21

```
4634 \bool_if:NT \c__mikoslides_notes_bool {
4635   \newcommand\pause{}
4636 }
```

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

`nomtext`

```
4637 \bool_if:NTF \c__mikoslides_notes_bool {
4638   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4639 }{
4640   \excludecomment{nomtext}
4641 }
```

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

```

nomgroup
4642 \bool_if:NTF \c__mikoslides_notes_bool {
4643   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4644 }{
4645   \excludecomment{nomgroup}
4646 }

ndefinition
4647 \bool_if:NTF \c__mikoslides_notes_bool {
4648   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4649 }{
4650   \excludecomment{ndefinition}
4651 }

nassertion
4652 \bool_if:NTF \c__mikoslides_notes_bool {
4653   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4654 }{
4655   \excludecomment{nassertion}
4656 }

nsproof
4657 \bool_if:NTF \c__mikoslides_notes_bool {
4658   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4659 }{
4660   \excludecomment{nsproof}
4661 }

nexample
4662 \bool_if:NTF \c__mikoslides_notes_bool {
4663   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4664 }{
4665   \excludecomment{nexample}
4666 }

\inputref@*skip We customize the hooks for in \inputref.
4667 \def\inputref@preskip{\smallskip}
4668 \def\inputref@postskip{\medskip}

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

\inputref*
4669 \let\orig@inputref\inputref
4670 \def\inputref{\@ifstar\ninputref\orig@inputref}
4671 \newcommand\ninputref[2] [] {
4672   \bool_if:NT \c__mikoslides_notes_bool {
4673     \orig@inputref[#1]{#2}
4674   }
4675 }

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

```

38.3 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>}`.

```

4676 \newlength{\slidelogoheight}
4677
4678 \bool_if:NTF \c__mikoslides_notes_bool {
4679   \setlength{\slidelogoheight}{.4cm}
4680 }{
4681   \setlength{\slidelogoheight}{1cm}
4682 }
4683 \newsavebox{\slidelogo}
4684 \sbox{\slidelogo}{\TeX}
4685 \newrobustcmd{\setslidelogo}[1]{
4686   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4687 }
```

(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.

```

4688 \def\source{Michael Kohlhase}% customize locally
4689 \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.

```

4690 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4691 \newsavebox{\cclogo}
4692 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4693 \newif\ifcchref\cchreffalse
4694 \AtBeginDocument{
4695   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4696 }
4697 \def\licensing{
4698   \ifcchref
4699     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4700   \else
4701     {\usebox{\cclogo}}
4702   \fi
4703 }
4704 \newrobustcmd{\setlicensing}[2][]{
4705   \def\@url{#1}
4706   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4707   \ifx\@url\@empty
4708     \def\licensing{{\usebox{\cclogo}}}
4709   \else
4710     \def\licensing{
```

```

4711     \ifcchref
4712     \href{#1}{\usebox{\cclogo}}
4713   \else
4714     {\usebox{\cclogo}}
4715   \fi
4716 }
4717 \fi
4718 }

```

(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

```

4719 \newrobustcmd\miko@slidelabel{
4720   \vbox to \slidelogoheight{
4721     \vss\hbox to \slidewidth
4722     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4723   }
4724 }

```

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

38.4 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.

```

4725 \def\Gin@mhrepos{}
4726 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4727 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4728 \newrobustcmd\frameimage[2][]{
4729   \stepcounter{slide}
4730   \bool_if:NT \c__mikoslides_frameimages_bool {
4731     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4732     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4733     \begin{center}
4734       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4735         \fbox{
4736           \ifx\Gin@ewidth\@empty
4737             \ifx\Gin@mhrepos\@empty
4738               \mhgraphics[width=\slidewidth,#1]{#2}
4739             \else
4740               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4741             \fi
4742           \else% Gin@ewidth empty
4743             \ifx\Gin@mhrepos\@empty
4744               \mhgraphics[#1]{#2}
4745             \else
4746               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4747             \fi
4748           \fi% Gin@ewidth empty
4749         }
4750       }{
4751         \ifx\Gin@ewidth\@empty

```

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


```

4752         \ifx\Gin@mhrepos\empty
4753             \mhgraphics[width=\slidewidth,#1]{#2}
4754         \else
4755             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4756         \fi
4757         \ifx\Gin@mhrepos\empty
4758             \mhgraphics[#1]{#2}
4759         \else
4760             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4761         \fi
4762     \fi% Gin@ewidth empty
4763 }
4764 \end{center}
4765 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4766 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4767 }
4768 } % ifmks@sty@frameimages

```

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

38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

4769 \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

```

4770 \AddToHook{begindocument}{
4771     \definecolor{green}{rgb}{0,.5,0}
4772     \definecolor{purple}{cmyk}{.3,1,0,.17}
4773 }

```

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`.

```

4774 % \def\STpresent#1{\textcolor{blue}{#1}}
4775 \def\defemph#1{\textcolor{magenta}{#1}}
4776 \def\symrefemph#1{\textcolor{cyan}{#1}}
4777 \def\compemph#1{\textcolor{blue}{#1}}
4778 \def\titleemph#1{\textcolor{blue}{#1}}
4779 \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.

```

4780 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4781 \def\smalltextwarning{
4782     \pgfuseimage{miko@small@dbend}
4783     \xspace
4784 }
4785 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4786 \newrobustcmd\textwarning{
4787   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4788   \xspace
4789 }
4790 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4791 \newrobustcmd\bigtextwarning{
4792   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4793   \xspace
4794 }

```

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

```

4795 \newrobustcmd\putgraphicsat[3]{
4796   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4797 }
4798 \newrobustcmd\putat[2]{
4799   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4800 }

```

38.6 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`.

```

4801 \bool_if:NT \c__mikoslides_sectocframes_bool {
4802   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4803     \newcounter{chapter}\counterwithin*{section}{chapter}
4804   }{
4805     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4806       \newcounter{chapter}\counterwithin*{section}{chapter}
4807     }
4808   }
4809 }

```

`\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
4810 \def\part@prefix{}
4811 \@ifpackageloaded{omdoc}{}{
4812   \str_case:VnF \__mikoslidestopsect {
4813     {part}{
4814       \int_set:Nn \l_document_structure_section_level_int {0}
4815       \def\thesection{\arabic{chapter}.\arabic{section}}
4816       \def\part@prefix{\arabic{chapter}.}
4817     }
4818     {chapter}{
4819       \int_set:Nn \l_document_structure_section_level_int {1}
4820       \def\thesection{\arabic{chapter}.\arabic{section}}
4821       \def\part@prefix{\arabic{chapter}.}
4822     }
4823   }{
4824     \int_set:Nn \l_document_structure_section_level_int {2}
4825     \def\part@prefix{}

```

```

4826 }
4827 }
4828
4829 \bool_if:NF \c__mikoslides_notes_bool { % only in slides

```

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

The new counters are used in the omgroup environment that choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

4830 \renewenvironment{omgroup}[2][]{
4831   \__document_structure_omgroup_args:n { #1 }
4832   \int_incr:N \l_document_structure_omgroup_level_int
4833   \int_incr:N \l_document_structure_section_level_int
4834   \bool_if:NT \c__mikoslides_sectocframes_bool {
4835     \stepcounter{slide}
4836     \begin{frame}[noframenumbering]
4837       \vfill\Large\centering
4838       \red{
4839         \ifcase\l_document_structure_section_level_int\or
4840           \stepcounter{part}
4841           \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4842           \def\currentsectionlevel{\omdoc@part@kw}
4843         \or
4844           \stepcounter{chapter}
4845           \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4846           \def\currentsectionlevel{\omdoc@chapter@kw}
4847         \or
4848           \stepcounter{section}
4849           \def\__mikoslideslabel{\part@prefix\arabic{section}}
4850           \def\currentsectionlevel{\omdoc@section@kw}
4851         \or
4852           \stepcounter{subsection}
4853           \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4854           \def\currentsectionlevel{\omdoc@subsection@kw}
4855         \or
4856           \stepcounter{subsubsection}
4857           \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4858           \def\currentsectionlevel{\omdoc@subsubsection@kw}
4859         \or
4860           \stepcounter{paragraph}
4861           \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
4862           \def\currentsectionlevel{\omdoc@paragraph@kw}
4863         \else
4864           \def\__mikoslideslabel{}
4865           \def\currentsectionlevel{\omdoc@paragraph@kw}
4866         \fi% end ifcase
4867         \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4868         \quad #2%
4869       }%
4870       \vfill%
4871       \end{frame}%
4872     }
4873     \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

4874 }{}
4875 }

```

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

```

4876 \def\inserttheorembodyfont{\normalfont}
4877 %\bool_if:NF \c__mikoslides_notes_bool {
4878 % \defbeamertemplate{theorem begin}{miko}
4879 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4880 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4881 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
4882 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

4883 % \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.

```

4884 % \expandafter\def\csname Parent2\endcsname{}
4885 %}
4886 \setbeamertemplate{theorems}[ams style]
4887 \bool_if:NT \c__mikoslides_notes_bool {
4888 \renewenvironment{columns}[1][\{
4889 \par\noindent%
4890 \begin{minipage}%
4891 \slidewidth\centering\leavevmode%
4892 }\{
4893 \end{minipage}\par\noindent%
4894 }\}
4895 \newsavebox\columnbox%
4896 \renewenvironment<>{column}[2][\{
4897 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4898 }\{
4899 \end{minipage}\end{lrbox}\usebox\columnbox%
4900 }\}
4901 }
4902 \bool_if:NTF \c__mikoslides_noproblems_bool {
4903 \newenvironment{problems}{}{}
4904 }{
4905 \excludecomment{problems}
4906 }

```

38.7 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.

```

4907 \gdef\printexcursions{}
4908 \newcommand\excursionref[2]{% label, text
4909 \bool_if:NT \c__mikoslides_notes_bool {
4910 \begin{sparagraph}[title=Excursion]
4911 #2 \sref[fallback=the appendix]{#1}.
4912 \end{sparagraph}

```

```

4913 }
4914 }
4915 \newcommand\activate@excursion[2][]{
4916   \gappto\printexcursions{\inputref{#1}{#2}}
4917 }
4918 \newcommand\excursion[4][]{% repos, label, path, text
4919   \bool_if:NT \c__mikoslides_notes_bool {
4920     \activate@excursion[1]{#3}\excursionref{#2}{#4}
4921   }
4922 }

```

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

\excursiongroup

```

4923 \keys_define:nn{mikoslides / excursiongroup }{
4924   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4925   intro       .tl_set:N    = \l__mikoslides_excursion_intro_tl,
4926   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4927 }
4928 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4929   \tl_clear:N \l__mikoslides_excursion_intro_tl
4930   \str_clear:N \l__mikoslides_excursion_id_str
4931   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4932   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4933 }
4934 \newcommand\excursiongroup[1][]{
4935   \__mikoslides_excursion_args:n{ #1 }
4936   \ifdefempty\printexcursions{}% only if there are excursions
4937   {\begin{note}
4938     \begin{omgroup}[#1]{Excursions}%
4939     \ifdefempty\l__mikoslides_excursion_intro_tl{\{
4940       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4941         \l__mikoslides_excursion_intro_tl
4942       }
4943     }
4944     \printexcursions%
4945     \end{omgroup}
4946   \end{note}}
4947 }
4948 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
4949 \</package>

```

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

Chapter 39

The Implementation

39.1 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.

```
4950 <*package>
4951 <@@=problems>
4952 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4953 \RequirePackage{l3keys2e,expl-keystr-compat}
4954
4955 \keys_define:nn { problem / pkg }{
4956   notes      .default:n    = { true },
4957   notes      .bool_set:N   = \c__problems_notes_bool,
4958   gnotes     .default:n    = { true },
4959   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4960   hints      .default:n    = { true },
4961   hints      .bool_set:N   = \c__problems_hints_bool,
4962   solutions  .default:n    = { true },
4963   solutions  .bool_set:N   = \c__problems_solutions_bool,
4964   pts        .default:n    = { true },
4965   pts        .bool_set:N   = \c__problems_pts_bool,
4966   min        .default:n    = { true },
4967   min        .bool_set:N   = \c__problems_min_bool,
4968   boxed      .default:n    = { true },
4969   boxed      .bool_set:N   = \c__problems_boxed_bool,
4970   unknown    .code:n       = {}
4971 }
4972 \def\solutionstrue{
4973   \bool_set_true:N \c__problems_solutions_bool
4974 }
4975 \def\solutionsfalse{
4976   \bool_set_false:N \c__problems_solutions_bool
4977 }
4978
4979 \ProcessKeysOptions{ problem / pkg }
```

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

```

4980 \RequirePackage{stex-compatibility}
4981 \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.

```

4982 \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.

```

4983 \def\prob@problem@kw{Problem}
4984 \def\prob@solution@kw{Solution}
4985 \def\prob@hint@kw{Hint}
4986 \def\prob@note@kw{Note}
4987 \def\prob@gnote@kw{Grading}
4988 \def\prob@pt@kw{pt}
4989 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4990 \@ifpackageloaded{babel}{
4991   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4992   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4993     \input{problem-ngerman.ldf}
4994   }
4995   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4996     \input{problem-finnish.ldf}
4997   }
4998   \clist_if_in:NnT \l_tmpa_clist {french}{
4999     \input{problem-french.ldf}
5000   }
5001   \clist_if_in:NnT \l_tmpa_clist {russian}{
5002     \input{problem-russian.ldf}
5003   }
5004 }{}

```

39.2 Problems and Solutions

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

```

5005 \keys_define:nn{ problem / problem }{
5006   id      .str_set:x:N = \l__problems_prob_id_str,
5007   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5008   min     .tl_set:N    = \l__problems_prob_min_tl,
5009   title   .tl_set:N    = \l__problems_prob_title_tl,
5010   refnum  .int_set:N   = \l__problems_prob_refnum_int
5011 }
5012 \cs_new_protected:Nn \__problems_prob_args:n {
5013   \str_clear:N \l__problems_prob_id_str
5014   \tl_clear:N \l__problems_prob_pts_tl
5015   \tl_clear:N \l__problems_prob_min_tl
5016   \tl_clear:N \l__problems_prob_title_tl

```

```

5017 \int_zero_new:N \l__problems_prob_refnum_int
5018 \keys_set:nn { problem / problem }{ #1 }
5019 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5020   \let\l__problems_inclprob_refnum_int\undefined
5021 }
5022 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5023 \newcounter{problem}
5024 \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.

```

5025 \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

```

5026 \newcommand\prob@number{
5027   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5028     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5029   }{
5030     \int_if_exist:NTF \l__problems_prob_refnum_int {
5031       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5032     }{
5033       \prob@label\theproblem
5034     }
5035   }
5036 }

```

(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.

```

5037 \newcommand\prob@title[3]{%
5038   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5039     #2 \l__problems_inclprob_title_tl #3
5040   }{
5041     \tl_if_exist:NTF \l__problems_prob_title_tl {
5042       #2 \l__problems_prob_title_tl #3
5043     }{
5044       #1
5045     }
5046   }
5047 }

```

(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.

```

5048 \def\prob@heading{
5049   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5050   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
5051 }

```

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

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`problem`

```

5052 \newenvironment{problem}[1][1]{
5053   \_problems_prob_args:n{#1}%\sref@target%
5054   \@in@omtexttrue% we are in a statement (for inline definitions)
5055   \stepcounter{problem}\record@problem
5056   \def\current@section@level{\prob@problem@kw}
5057   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5058 }%
5059   {\smallskip}
5060   \bool_if:NT \c__problems_boxed_bool {
5061     \surroundwithmdframed{problem}
5062   }

```

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

```

5063 \def\record@problem{
5064   \protected@write\@auxout{}
5065   {
5066     \string\@problem{\prob@number}
5067     {
5068       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5069         \l__problems_inclprob_pts_tl
5070       }{
5071         \l__problems_prob_pts_tl
5072       }
5073     }%
5074     {
5075       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5076         \l__problems_inclprob_min_tl
5077       }{
5078         \l__problems_prob_min_tl
5079       }
5080     }
5081   }
5082 }

```

(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).

```

5083 \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.

```

5084 \keys_define:nn { problem / solution }{
5085   id          .str_set_x:N = \l__problems_solution_id_str ,
5086   for         .tl_set:N    = \l__problems_solution_for_tl ,
5087   height      .dim_set:N   = \l__problems_solution_height_dim ,
5088   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5089   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5090   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5091 }
5092 \cs_new_protected:Nn \__problems_solution_args:n {
5093   \str_clear:N \l__problems_solution_id_str
5094   \tl_clear:N \l__problems_solution_for_tl
5095   \tl_clear:N \l__problems_solution_srccite_tl
5096   \clist_clear:N \l__problems_solution_creators_clist
5097   \clist_clear:N \l__problems_solution_contributors_clist
5098   \dim_zero:N \l__problems_solution_height_dim
5099   \keys_set:nn { problem / solution }{ #1 }
5100 }

```

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

```

5101 \newcommand\@startsolution[1][ ]{
5102   \__problems_solution_args:n { #1 }
5103   \@in@omtexttrue% we are in a statement.
5104   \bool_if:NF \c__problems_boxed_bool { \hrule }
5105   \smallskip\noindent
5106   {\textbf\prob@solution@kw : \enspace}
5107   \begin{small}
5108   \def\current@section@level{\prob@solution@kw}
5109   \ignorespacesandpars
5110 }

```

\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.

```

5111 \newcommand\startsolutions{
5112   \specialcomment{solution}{\@startsolution}{
5113     \bool_if:NF \c__problems_boxed_bool {
5114       \hrule\medskip
5115     }
5116     \end{small}%
5117   }
5118   \bool_if:NT \c__problems_boxed_bool {
5119     \surroundwithmdframed{solution}
5120   }
5121 }

```

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

\stopsolutions

```

5122 \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.

```

5123 \bool_if:NTF \c__problems_solutions_bool {
5124   \startsolutions
5125 }{
5126   \stopsolutions
5127 }

```

exnote

```

5128 \bool_if:NTF \c__problems_notes_bool {
5129   \newenvironment{exnote}[1][]{
5130     \par\smallskip\hrule\smallskip
5131     \noindent\textbf{\prob@note@kw : }\small
5132   }{
5133     \smallskip\hrule
5134   }
5135 }{
5136   \excludecomment{exnote}
5137 }

```

hint

```

5138 \bool_if:NTF \c__problems_notes_bool {
5139   \newenvironment{hint}[1][]{
5140     \par\smallskip\hrule\smallskip
5141     \noindent\textbf{\prob@hint@kw :~ }\small
5142   }{
5143     \smallskip\hrule
5144   }
5145   \newenvironment{exhint}[1][]{
5146     \par\smallskip\hrule\smallskip
5147     \noindent\textbf{\prob@hint@kw :~ }\small
5148   }{
5149     \smallskip\hrule
5150   }
5151 }{
5152   \excludecomment{hint}
5153   \excludecomment{exhint}
5154 }

```

gnote

```

5155 \bool_if:NTF \c__problems_notes_bool {
5156   \newenvironment{gnote}[1][]{
5157     \par\smallskip\hrule\smallskip
5158     \noindent\textbf{\prob@gnote@kw : }\small
5159   }{
5160     \smallskip\hrule
5161   }
5162 }{
5163   \excludecomment{gnote}
5164 }

```

39.3 Multiple Choice Blocks

EdN:23

mcb 23

```

5165 \newenvironment{mcb}{
5166   \begin{enumerate}
5167 }{
5168   \end{enumerate}
5169 }
```

we define the keys for the mcc macro

```

5170 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5171   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5172     \bool_set_true:N #1
5173   }{
5174     \bool_set_false:N #1
5175   }
5176 }
5177 \keys_define:nn { problem / mcc }{
5178   id          .str_set:x:N = \l__problems_mcc_id_str ,
5179   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5180   T           .default:n    = { true } ,
5181   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5182   F           .default:n    = { true } ,
5183   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5184   Ttext       .code:n       = {
5185     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5186   } ,
5187   Ftext       .code:n       = {
5188     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5189   }
5190 }
5191 \cs_new_protected:Nn \l__problems_mcc_args:n {
5192   \str_clear:N \l__problems_mcc_id_str
5193   \tl_clear:N \l__problems_mcc_feedback_tl
5194   \bool_set_true:N \l__problems_mcc_t_bool
5195   \bool_set_true:N \l__problems_mcc_f_bool
5196   \bool_set_true:N \l__problems_mcc_Ttext_bool
5197   \bool_set_false:N \l__problems_mcc_Ftext_bool
5198   \keys_set:nn { problem / mcc }{ #1 }
5199 }
```

\mcc

```

5200 \newcommand\mcc[2][]{
5201   \l__problems_mcc_args:n{ #1 }
5202   \item #2
5203   \bool_if:NT \c__problems_solutions_bool {
5204     \\\
5205     \bool_if:NT \l__problems_mcc_t_bool {
5206       % TODO!
5207       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5208     }
5209     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

5210      % TODO!
5211      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5212    }
5213    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5214      !
5215    }{
5216      \l__problems_mcc_feedback_tl
5217    }
5218  }
5219 } %solutions

```

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

39.4 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.

```

5220
5221 \keys_define:nn{ problem / inclproblem }{
5222   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5223   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5224   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5225   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5226   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5227   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5228 }
5229
5230 \cs_new_protected:Nn \__problems_inclprob_args:n {
5231   % \str_clear:N \l__problems_prob_id_str
5232   \tl_clear:N \l__problems_inclprob_pts_tl
5233   \tl_clear:N \l__problems_inclprob_min_tl
5234   \tl_clear:N \l__problems_inclprob_title_tl
5235   \int_zero_new:N \l__problems_inclprob_refnum_int
5236   \str_clear:N \l__problems_inclprob_mhrepos_str
5237   \keys_set:nn { problem / inclproblem }{ #1 }
5238   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5239     \let\l__problems_inclprob_pts_tl\undefined
5240   }
5241   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5242     \let\l__problems_inclprob_min_tl\undefined
5243   }
5244   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5245     \let\l__problems_inclprob_title_tl\undefined
5246   }
5247   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5248     \let\l__problems_inclprob_refnum_int\undefined
5249   }
5250 }
5251
5252 \cs_new_protected:Nn \__problems_inclprob_clear: {
5253   % \str_clear:N \l__problems_prob_id_str
5254   \let\l__problems_inclprob_pts_tl\undefined
5255   \let\l__problems_inclprob_min_tl\undefined

```

```

5255 \let\l__problems_inclprob_title_tl\undefined
5256 \let\l__problems_inclprob_refnum_int\undefined
5257 \let\l__problems_inclprob_mhrepos_str\undefined
5258 }
5259
5260 \newcommand\includeproblem[2][]{
5261   \__problems_inclprob_args:n{ #1 }
5262   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5263     \input{#2}
5264   }{
5265     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5266       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5267     }
5268   }
5269   \__problems_inclprob_clear:
5270 }

```

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

39.5 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.

```

5271 \AddToHook{enddocument}{
5272   \bool_if:NT \c__problems_pts_bool {
5273     \message{Total:~\arabic{pts}~points}
5274   }
5275   \bool_if:NT \c__problems_min_bool {
5276     \message{Total:~\arabic{min}~minutes}
5277   }
5278 }

```

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

```

5279 \def\pts#1{
5280   \bool_if:NT \c__problems_pts_bool {
5281     \marginpar{#1~\prob@pt@kw}
5282   }
5283 }
5284 \def\min#1{
5285   \bool_if:NT \c__problems_min_bool {
5286     \marginpar{#1~\prob@min@kw}
5287   }
5288 }

```

\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.

```

5289 \newcounter{pts}
5290 \def\show@pts{
5291   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5292     \bool_if:NT \c__problems_pts_bool {
5293       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5294       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5295     }
5296   }{
5297     \tl_if_exist:NT \l__problems_prob_pts_tl {
5298       \bool_if:NT \c__problems_pts_bool {
5299         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5300         \addtocounter{pts}{\l__problems_prob_pts_tl}
5301       }
5302     }
5303   }
5304 }

```

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

\show@min

```

5305 \newcounter{min}
5306 \def\show@min{
5307   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5308     \bool_if:NT \c__problems_min_bool {
5309       \marginpar{\l__problems_inclprob_pts_tl;min}
5310       \addtocounter{min}{\l__problems_inclprob_min_tl}
5311     }
5312   }{
5313     \tl_if_exist:NT \l__problems_prob_min_tl {
5314       \bool_if:NT \c__problems_min_bool {
5315         \marginpar{\l__problems_prob_min_tl;min}
5316         \addtocounter{min}{\l__problems_prob_min_tl}
5317       }
5318     }
5319   }
5320 }
5321 \</package>

```

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

Chapter 40

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.

40.1 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.

```
5322 <@@=hwexam>
5323 <*cls>
5324 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5325 \RequirePackage{l3keys2e,expl-keystr-compatible}
5326 \DeclareOption*{
5327   \PassOptionsToClass{\CurrentOption}{omdoc}
5328   \PassOptionsToPackage{\CurrentOption}{stex}
5329   \PassOptionsToPackage{\CurrentOption}{hwexam}
5330   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5331 }
5332 \ProcessOptions
```

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

```
5333 \LoadClass{omdoc}
5334 \RequirePackage{stex}
5335 \RequirePackage{hwexam}
5336 \RequirePackage{tikzinput}
5337 \RequirePackage{graphicx}
5338 \RequirePackage{a4wide}
5339 \RequirePackage{amssymb}
5340 \RequirePackage{amstext}
5341 \RequirePackage{amsmath}
```

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


```

5342 \newcommand\assig@default@type{\hwexam@assignment@kw}
5343 \def\document@hwexamtype{\assig@default@type}
5344 <@@=document_structure>
5345 \keys_define:nn { document-structure / document }{
5346 id .str_set_x:N = \c_document_structure_document_id_str,
5347 hwexamtype .tl_set:N = \document@hwexamtype
5348 }
5349 <@@=hwexam>
5350 </cls>

```

Chapter 41

Implementation: The hwexam Package

41.1 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.

```
5351 \*package>
5352 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5353 \RequirePackage{l3keys2e,expl-keystr-compat}
5354
5355 \newif\iftest\testfalse
5356 \DeclareOption{test}{\testtrue}
5357 \newif\ifmultiple\multiplefalse
5358 \DeclareOption{multiple}{\multipletrue}
5359 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5360 \ProcessOptions
```

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

```
5361 \RequirePackage{keyval}[1997/11/10]
5362 \RequirePackage{problem}
```

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

```
5363 \newcommand\hwexam@assignment@kw{Assignment}
5364 \newcommand\hwexam@given@kw{Given}
5365 \newcommand\hwexam@due@kw{Due}
5366 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5367   space}%
5368 \newcommand\correction@probs@kw{prob.}%
5369 \newcommand\correction@pts@kw{total}%
5370 \newcommand\correction@reached@kw{reached}%
5371 \newcommand\correction@sum@kw{Sum}%
5372 \newcommand\correction@grade@kw{grade}%
5373 \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

```

5374 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5375
5376 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5377 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5378   \input{hwexam-ngerman.lda}
5379 }
5380 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5381   \input{hwexam-finnish.lda}
5382 }
5383 \clist_if_in:NnT \l_tmpa_clist {french}{
5384   \input{hwexam-french.lda}
5385 }
5386 \clist_if_in:NnT \l_tmpa_clist {russian}{
5387   \input{hwexam-russian.lda}
5388 }

```

41.2 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.

```

5389 \newcounter{assignment}
5390 \numberproblemsin{assignment}
5391 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5392 \keys_define:nn { hwexam / assignment } {
5393   id .str_set:N = \l__hwexam_assign_id_str,
5394   number .int_set:N = \l__hwexam_assign_number_int,
5395   title .tl_set:N = \l__hwexam_assign_title_tl,
5396   type .tl_set:N = \l__hwexam_assign_type_tl,
5397   given .tl_set:N = \l__hwexam_assign_given_tl,
5398   due .tl_set:N = \l__hwexam_assign_due_tl,
5399   loadmodules .code:n = {
5400     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5401   }
5402 }
5403 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5404   \str_clear:N \l__hwexam_assign_id_str
5405   \int_set:Nn \l__hwexam_assign_number_int {-1}
5406   \tl_clear:N \l__hwexam_assign_title_tl
5407   \tl_clear:N \l__hwexam_assign_type_tl
5408   \tl_clear:N \l__hwexam_assign_given_tl
5409   \tl_clear:N \l__hwexam_assign_due_tl
5410   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5411   \keys_set:nn { hwexam / assignment }{ #1 }
5412 }

```

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.

```

5413 \newcommand\given@due[2]{
5414 \bool_lazy_all:nF {
5415 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5416 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5417 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5418 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5419 }{ #1 }
5420
5421 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5422 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5423 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5424 }
5425 }{
5426 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5427 }
5428
5429 \bool_lazy_or:nnF {
5430 \bool_lazy_and_p:nn {
5431 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5432 }{
5433 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5434 }
5435 }{
5436 \bool_lazy_and_p:nn {
5437 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5438 }{
5439 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5440 }
5441 }{ ,~ }
5442
5443 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5444 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5445 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5446 }
5447 }{
5448 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5449 }
5450
5451 \bool_lazy_all:nF {
5452 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5453 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5454 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5455 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5456 }{ #2 }
5457 }

```

`\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.

```

5458 \newcommand\assignment@title[3]{

```

```

5459 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5460 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5461 #1
5462 }{
5463 #2\l__hwexam_assign_title_tl#3
5464 }
5465 }{
5466 #2\l__hwexam_inclassassign_title_tl#3
5467 }
5468 }

```

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

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

```

5469 \newcommand\assignment@number{
5470 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5471 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5472 \int_use:N \l__hwexam_assign_number_int
5473 }
5474 }{
5475 \int_use:N \l__hwexam_inclassassign_number_int
5476 }
5477 }

```

(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.

```

5478 \newenvironment{assignment}[1][]{
5479 \__hwexam_assignment_args:n { #1 }
5480 %\sref@target
5481 \let\__hwexamnum\l__hwexam_assign_number_int
5482 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5483 \stepcounter{assignment}
5484 }{
5485 \setcounter{assignment}{\int_use:N\__hwexamnum}
5486 }
5487 \setcounter{problem}{0}
5488 \def\current@section@level{\document@hwexamtype}
5489 %\sref@label@id{\document@hwexamtype \thesection}
5490 \begin{@assignment}
5491 }{
5492 \end{@assignment}
5493 }

```

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

```

5494 \def\__hwexasstitle{
5495 \protect\document@hwexamtype~\arabic{assignment}
5496 \assignment@title{}\;{} \; -- \given@due{}{}
5497 }

```

```

5498 \ifmultiple
5499 \newenvironment{@assignment}{
5500 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5501 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5502 }{
5503 \begin{omgroup}{\__hwexasstitle}
5504 }
5505 }{
5506 \end{omgroup}
5507 }

```

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

```

5508 \else
5509 \newenvironment{@assignment}{
5510 \begin{center}\bf
5511 \Large\@title\strut\
5512 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}\}
5513 \large\given@due{--\;}{\;}{--}
5514 \end{center}
5515 }{}
5516 \fi% multiple

```

41.3 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.

```

5517 \keys_define:nn { hwexam / inclassignment } {
5518 %id .str_set_x:N = \l__hwexam_assign_id_str,
5519 number .int_set:N = \l__hwexam_inclassign_number_int,
5520 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5521 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5522 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5523 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5524 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5525 }
5526 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5527 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5528 \tl_clear:N \l__hwexam_inclassign_title_tl
5529 \tl_clear:N \l__hwexam_inclassign_type_tl
5530 \tl_clear:N \l__hwexam_inclassign_given_tl
5531 \tl_clear:N \l__hwexam_inclassign_due_tl
5532 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5533 \keys_set:nn { hwexam / inclassignment }{ #1 }
5534 }
5535 \__hwexam_inclassignment_args:n {}
5536
5537 \newcommand\inputassignment[2][ ]{
5538 \__hwexam_inclassignment_args:n { #1 }
5539 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5540 \input{#2}
5541 }{
5542 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5543 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5544 }
5545 }
5546 \__hwexam_inclasssignment_args:n {}
5547 }
5548 \newcommand\includeassignment[2][ ]{
5549 \newpage
5550 \inputassignment[#1]{#2}
5551 }

```

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

41.4 Typesetting Exams

\quizheading

```

5552 \ExplSyntaxOff
5553 \newcommand\quizheading[1]{%
5554 \def\@tas{#1}%
5555 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5556 \ifx\@tas\empty\else%
5557 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5558 \fi%
5559 }
5560 \ExplSyntaxOn

```

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

\testheading

```

5561 \keys_define:nn { hwexam / testheading } {
5562 min .tl_set:N = \l__hwexam_testheading_min_tl,
5563 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5564 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5565 }
5566 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5567 \tl_clear:N \l__hwexam_testheading_min_tl
5568 \tl_clear:N \l__hwexam_testheading_duration_tl
5569 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5570 \keys_set:nn { hwexam / testheading }{ #1 }
5571 }
5572 \newenvironment{testheading}[1][ ]{
5573 \__hwexam_testheading_args:n{ #1 }
5574 \noindent\large{Name:~\hfill
5575 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5576 \begin{center}
5577 \Large\textbf{\@title}\[1ex]
5578 \large\@date\[3ex]
5579 \end{center}
5580 \textbf{You~have~
5581 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5582 \l__hwexam_testheading_min_tl~minutes
5583 }{
5584 \l__hwexam_testheading_duration_tl
5585 }~

```

```

5586 (sharp)~for~the~test
5587 };\
5588 Write~the~solutions~to~the~sheet.
5589 \par\noindent
5590 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5591 \advance\check@time by -\theassignment@totalmin
5592 The~estimated~time~for~solving~this~exam~is~
5593 {\theassignment@totalmin}~minutes,~
5594 leaving~you~{\the\check@time}~minutes~for~revising~
5595 your~exam.
5596
5597 \par\noindent
5598 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5599 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5600 You~can~reach~{\theassignment@totalpts}~points~if~you~
5601 solve~all~problems.~You~will~only~need~
5602 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5603 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
5604 \vfill
5605 \begin{center}
5606 {
5607 \Large\em You~have~ample~time,~so~take~it~slow~
5608 and~avoid~rushing~to~mistakes!\}[2ex]
5609 Different~problems~test~different~skills~and~
5610 knowledge,~so~do~not~get~stuck~on~one~problem.
5611 }
5612 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5613 \end{center}
5614 }{
5615 \newpage
5616 }

```

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

\testspace

```

5617 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5618 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5619 \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.

```

5620 <@=problems>
5621 \renewcommand\@problem[3]{
5622 \stepcounter{assignment@probs}
5623 \def\__problemspts{#2}

```



```

5624 \ifx\__problemspts\@empty\else
5625 \addtocounter{assignment@totalpts}{#2}
5626 \fi
5627 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5628 \xdef\correction@probs{\correction@probs & #1}%
5629 \xdef\correction@pts{\correction@pts & #2}
5630 \xdef\correction@reached{\correction@reached &}
5631 }
5632 \@@=hwexam

```

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

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

```

5633 \newcounter{assignment@probs}
5634 \newcounter{assignment@totalpts}
5635 \newcounter{assignment@totalmin}
5636 \def\correction@probs{\correction@probs@kw}%
5637 \def\correction@pts{\correction@pts@kw}%
5638 \def\correction@reached{\correction@reached@kw}%
5639 \def\after@correction@table{}%
5640 \stepcounter{assignment@probs}
5641 \newcommand\correction@table{
5642 \resizebox{\textwidth}{!}{%
5643 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5644 &\multicolumn{\theassignment@probs}{c|}||%|
5645 {\footnotesize\correction@forgrading@kw} &\\ \hline
5646 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5647 \correction@pts & \theassignment@totalpts & \\ \hline
5648 \correction@reached & & \[.7cm]\hline
5649 \end{tabular}}
5650 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5651 \end{package}

```

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

41.5 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\wierfont=../assignments/wierfont
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\wierfont{\font\wierfont\char65}
\newcommand\denkerfont{\font\denkerfont\char65}
\newcommand\uhrfont{\font\uhrfont\char65}
\newcommand\warnschildfont{\font\warnschildfont\char 65}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\wierfont}

```