

The sTeX3 Package *

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

2022-03-04

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 2022-03-04)

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
3	Using sTeX	5
3.1	How Knowledge is Organized in sTeX	5
3.2	A First sTeX Document	6
4	sTeX Archives	8
4.1	The Local MathHub-Directory	8
4.2	The Structure of sTeX Archives	8
4.3	MANIFEST.MF-Files	9
5	Creating New Modules and Symbols	10
5.1	Advanced Structuring Mechanisms	10
5.2	Primitive Symbols (The sTeX Metatheory)	12
6	sTeX Statements (Definitions, Theorems, Examples, ...)	13
7	Additional Packages	14
7.1	Modular Document Structuring	14
7.2	Slides and Course Notes	14
7.3	Homework, Problems and Exams	14
8	Stuff	15
8.1	Modules	15
8.1.1	Semantic Macros and Notations	15
	Other Argument Types	17
	Precedences	19
8.1.2	Archives and Imports	19
	Namespaces	19
	Paths in Import-Statements	20
II	Documentation	21
9	sTeX-Basics	22
9.1	Macros and Environments	22
9.1.1	HTML Annotations	22
9.1.2	Babel Languages	23
9.1.3	Auxiliary Methods	23

10	<code>sTeX</code>-MathHub	24
10.1	Macros and Environments	24
10.1.1	Files, Paths, URIs	24
10.1.2	MathHub Archives	25
10.1.3	Using Content in Archives	26
11	<code>sTeX</code>-References	27
11.1	Macros and Environments	27
11.1.1	Setting Reference Targets	27
11.1.2	Using References	28
12	<code>sTeX</code>-Modules	29
12.1	Macros and Environments	29
12.1.1	The <code>smodule</code> environment	31
13	<code>sTeX</code>-Module Inheritance	33
13.1	Macros and Environments	33
13.1.1	SMS Mode	33
13.1.2	Imports and Inheritance	34
14	<code>sTeX</code>-Symbols	36
14.1	Macros and Environments	36
15	<code>sTeX</code>-Terms	38
15.1	Macros and Environments	38
16	<code>sTeX</code>-Structural Features	40
16.1	Macros and Environments	40
16.1.1	Structures	40
17	<code>sTeX</code>-Statements	41
17.1	Macros and Environments	41
18	<code>sTeX</code>-Proofs: Structural Markup for Proofs	42
18.1	Introduction	44
18.2	The User Interface	45
18.2.1	Package Options	45
18.2.2	Proofs and Proof steps	45
18.2.3	Justifications	45
18.2.4	Proof Structure	47
18.2.5	Proof End Markers	47
18.2.6	Configuration of the Presentation	47
18.3	Limitations	48
19	<code>sTeX</code>-Metatheory	49
19.1	Symbols	49
III	Extensions	50

20	Tikzinput	51
20.1	Macros and Environments	51
21	document-structure: Semantic Markup for Open Mathematical Documents in \LaTeX	52
21.1	Introduction	52
21.2	The User Interface	53
21.2.1	Package and Class Options	53
21.2.2	Document Structure	53
21.2.3	Ignoring Inputs	55
21.2.4	Structure Sharing	55
21.2.5	Global Variables	55
21.2.6	Colors	56
21.3	Limitations	56
22	NotesSlides – Slides and Course Notes	57
22.1	Introduction	57
22.2	The User Interface	57
22.2.1	Package Options	57
22.2.2	Notes and Slides	58
22.2.3	Header and Footer Lines of the Slides	59
22.2.4	Frame Images	59
22.2.5	Colors and Highlighting	60
22.2.6	Front Matter, Titles, etc.	60
22.2.7	Excursions	60
22.2.8	Miscellaneous	61
22.3	Limitations	61
23	problem.sty: An Infrastructure for formatting Problems	62
23.1	Introduction	62
23.2	The User Interface	62
23.2.1	Package Options	62
23.2.2	Problems and Solutions	63
23.2.3	Multiple Choice Blocks	64
23.2.4	Including Problems	64
23.2.5	Reporting Metadata	64
23.3	Limitations	64
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	66
24.1	Introduction	67
24.2	The User Interface	67
24.2.1	Package and Class Options	67
24.2.2	Assignments	67
24.2.3	Typesetting Exams	67
24.2.4	Including Assignments	68
24.3	Limitations	68
IV	Implementation	70

25	gTeX-Basics Implementation	71
25.1	The gTeXDocument Class	71
25.2	Preliminaries	71
25.3	Messages and logging	72
25.4	HTML Annotations	73
25.5	Babel Languages	76
25.6	Auxiliary Methods	77
26	gTeX-MathHub Implementation	78
26.1	Generic Path Handling	78
26.2	PWD and kpsewhich	80
26.3	File Hooks and Tracking	81
26.4	MathHub Repositories	82
26.5	Using Content in Archives	86
27	gTeX-References Implementation	91
27.1	Document URIs and URLs	91
27.2	Setting Reference Targets	93
27.3	Using References	95
28	gTeX-Modules Implementation	98
28.1	The smodule environment	102
28.2	Invoking modules	107
29	gTeX-Module Inheritance Implementation	109
29.1	SMS Mode	109
29.2	Inheritance	112
30	gTeX-Symbols Implementation	117
30.1	Symbol Declarations	117
30.2	Notations	123
30.3	Variables	133
31	gTeX-Terms Implementation	139
31.1	Symbol Invocations	139
31.2	Terms	146
31.3	Notation Components	150
31.4	Variables	152
31.5	Sequences	154
32	gTeX-Structural Features Implementation	155
32.1	Imports with modification	156
32.2	The feature environment	162
32.3	Structure	163
33	gTeX-Statements Implementation	172
33.1	Definitions	172
33.2	Assertions	177
33.3	Examples	180
33.4	Logical Paragraphs	183

34 The Implementation	188
34.1 Package Options	188
34.2 Proofs	188
34.3 Justifications	199
35 \LaTeX-Others Implementation	201
36 \LaTeX-Metatheory Implementation	202
37 Tikzinput Implementation	205
38 document-structure.sty Implementation	207
38.1 The document-structure Class	207
38.2 Class Options	207
38.3 Beefing up the <code>document</code> environment	208
38.4 Implementation: document-structure Package	208
38.5 Package Options	208
38.6 Document Structure	210
38.7 Front and Backmatter	213
38.8 Global Variables	215
39 NotesSlides – Implementation	216
39.1 Class and Package Options	216
39.2 Notes and Slides	218
39.3 Header and Footer Lines	222
39.4 Frame Images	223
39.5 Colors and Highlighting	224
39.6 Sectioning	225
39.7 Excursions	228
40 The Implementation	229
40.1 Package Options	229
40.2 Problems and Solutions	230
40.3 Multiple Choice Blocks	236
40.4 Including Problems	237
40.5 Reporting Metadata	238
41 Implementation: The hwexam Class	240
41.1 Class Options	240
42 Implementation: The hwexam Package	242
42.1 Package Options	242
42.2 Assignments	243
42.3 Including Assignments	246
42.4 Typesetting Exams	247
42.5 Leftovers	249

Part I

Manual



Boxes like this one contain implementation details that are not immediately interesting, but might be useful for debugging errors.



Boxes like this one explain how some \LaTeX concept relates to the MMT/OMDoc system, philosophy or language.

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

- **The sTeX-Package** available [here](#)¹. Note, that the CTAN repository for L^AT_EX packages may contain outdated versions of the sTeX package, so make sure, that your TEXMF system variable is configured such that the packages available in the linked repository are prioritized over potential default packages that come with your T_EX distribution.
- To make sure that sTeX too knows where to find its archives, we need to set a global system variable MATHHUB, that points to your local MathHub-directory (see [chapter 4](#)).

- **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.

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

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

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

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

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

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

Chapter 3

Using sTeX

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the `standalone` document class with the `stex` package.

Both the `stex` package and document class offer the following options:

lang (*<language>**) Languages to load with the `babel` package.

mathhub (*<directory>*) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

sms (*<boolean>*) use *persisted* mode (not yet implemented).

image (*<boolean>*) passed on to `tikzinput`.

debug (*<log-prefix>**) Logs debugging information with the given prefixes to the terminal, or all if `all` is given. Largely irrelevant for the majority of users.

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

- sTeX **archives** (see [chapter 4](#)) contain individual `.tex`-files.
- These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
- Modules contain sTeX **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
- sTeX **expressions** finally are built up from usages of semantic macros.

$\text{\texttt{\textbackslash importmodules}}$ (and similar constructions) induce an MMT *includes* and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense.

$\text{\texttt{\textbackslash T}}$ Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.

Finally, $\text{\texttt{\textbackslash T}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

$\text{\texttt{\textbackslash importmodules}}$ (and similar constructions) induce an MMT *includes* and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense.

$\text{\texttt{\textbackslash T}}$ Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.

Finally, $\text{\texttt{\textbackslash T}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

$\text{\texttt{\textbackslash importmodules}}$ (and similar constructions) induce an MMT *includes* and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense.

$\text{\texttt{\textbackslash T}}$ Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.

Finally, $\text{\texttt{\textbackslash T}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

3.2 A First `TeX` Document

Having set everything up, we can write a first `TeX` document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated `MathHub`-folder, and write a small fragment defining the *geometric series*:

```

1 \documentclass{article}
2 \usepackage{stex,xcolor,stextmh}
3
4 \begin{document}
5   \begin{smodule}{GeometricSeries}
6     \importmodule[smglom/calculus]{series}
7     \importmodule[smglom/arithmetics]{realarith}
8
9     \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11     \begin{sdefinition}[for=geometricSeries]
12       The \definame{geometricSeries} is the \symname{?series}
13       \[ \infinitiesum{n}{1}{
14         \realdivide[frac]{1}{
15           \realpower{2}{n}
16         }
17       } . \]
18     \end{sdefinition}
19
20     \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
21       The \symname{geometricSeries} \symname{converges} towards  $\$1$ .
22     \end{sassertion}
23   \end{smodule}
24 \end{document}

```

Compiling this document with `pdflatex` should yield the output

Definition 0.1. The **geometric series** is the series

$$\sum_{n=1}^{\infty} \frac{1}{2^n}.$$

Theorem 0.2. The geometric series converges towards 1.

Here is what is happening under the hood in this document:

`smodule` First, we open a new *module* called `GeometricSeries`.

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 \TeX *symbols* formally declared somewhere; associating their *presen-*
tation 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: terms documentation

TODO: references documentation

³EdNOTE: somewhere later

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

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

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

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

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

TODO: modules documentation
TODO: symbols documentation
TODO: inheritance documentation

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

```

1 \begin{smodule}{magma}
2   \symdef{universe}{\comp{\mathcal U}}
3   \symdef{operation}[args=2,op=\circ]{\#1 \comp\circ \#2}
4 \end{smodule}
5 \begin{smodule}{monoid}
6   \importmodule{magma}
7   \symdef{unit}{\comp e}
8 \end{smodule}
9 \begin{smodule}{group}
10  \importmodule{monoid}
11  \symdef{inverse}[args=1]{\#1^{\comp{-1}}}
12 \end{smodule}

```

Module 2:

Module 3:

Module 4:

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

Example 3

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \renamedekl[name=universe]{universe}{runiverse}
4     \renamedekl[name=plus]{operation}{rplus}
5     \renamedekl[name=zero]{unit}{rzero}
6     \renamedekl[name=uminus]{inverse}{ruminus}
7   \end{copymodule}
8   \notation*{rplus}[plus,op=+,prec=60]{\#1 \comp+ \#2}
9   \notation*{rzero}[zero]{\comp0}
10  \notation*{ruminus}[uminus,op=-]{\comp- \#1}
11  \begin{copymodule}{monoid}{multiplication}
12    \assign{universe}{\runiverse}
13    \renamedekl[name=times]{operation}{rtimes}
14    \renamedekl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{\#1 \comp\cdot \#2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Module 5:

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

TODO: explain donotclone

Example 4

```
1 \begin{smodule}{int}
2   \symdef{Integers}{\comp{\mathbb Z}}
3   \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
4   \symdef{zero}{\comp0}
5   \symdef{uminus}[args=1,op=-]{\comp-#1}
6
7   \begin{interpretmodule}{group}{intisgroup}
8     \assign{universe}{\Integers}
9     \assign{operation}{\plus!}
10    \assign{unit}{\zero}
11    \assign{inverse}{\uminus!}
12  \end{interpretmodule}
13 \end{smodule}
```

Module 6:

5.2 Primitive Symbols (The \TeX Metatheory)

TODO: metatheory documentation

Chapter 6

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

TODO: statements documentation
TODO: sproofs documentation

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

Chapter 8

Stuff

8.1 Modules

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

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 5

```
1 \symdecl{mult}[args=2]
2 \notation{mult}{#1 #2}
3 \ult{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{mult}[args=2]{#1 #2}
```

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

Example 6

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

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

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

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

Example 7

```
1 \mult*{\arg{a}\comp{\ast}\arg{b}}$ is the
2 \lt{\comp{product of} \arg{$a$} \comp{and} \arg{$b$}}
```

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

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

Example 8

```
1 \ult{\comp{Multiplying} \arg*{$\mult{a}{b}$} again by \arg{$b$}} yields...
```

Multiplying again by b yields...

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

Example 9

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

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

⁴EdNOTE: TODO

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

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

Example 10

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

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

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

Example 11

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

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

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

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

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

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

There’s also a 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{forevery}[args=bi]{\forall #1.\; #2}
```

Module 8: b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDoc and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 12

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

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

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

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

Example 13

```
1 \ymdef{numseq}[args=ai]{#1 \comp\in #2}{##1 \comp\leq ##2}
2 \umseq{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 _x \int _y \int _z 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{foo}[prec=200;500x600]{#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.

$\text{\texttt{\textsf{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}}$ 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:

Module 9:

Example 14

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

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

8.1.2 Archives and Imports

Namespaces

Ideally, $\text{\texttt{\textsf{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}}$ 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, $\text{\texttt{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}}$ 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 $\text{\texttt{\textsf{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}}$ can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 9

sTeX-Basics

This sub package provides general set up code, auxiliary methods and abstractions for xhtml annotations.

9.1 Macros and Environments

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

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

Logs *<message>*, if the package option `debug` contains *<log-prefix>*.

9.1.1 HTML Annotations

<code>\if@latexml</code>	L ^A T _E X2e conditional for L ^A T _E XML
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if:TF *</code>	

<code>\stex_if_do_html_p: *</code>	Whether to currently produce any HTML annotations (can be false in some advanced structuring environments, for example)
<code>\stex_if_do_html:TF *</code>	

<code>\stex_suppress_html:n</code>	Temporarily disables HTML annotations in its argument code
------------------------------------	--

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

9.1.2 Babel Languages

<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`.

9.1.3 Auxiliary Methods

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

Makes the macro `⟨cs⟩` throw an error, indicating that it is only allowed in the context of `⟨environments⟩`.

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

<code>\ignorespacesandpars</code>	ignores white space characters and <code>\par</code> control sequences. Expands tokens in the process.
-----------------------------------	--

Chapter 10

STEX-MathHub

This sub package provides code for handling ST_EX archives, files, file paths and related methods.

10.1 Macros and Environments

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

10.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{\langle string \rangle\}$ turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$. Also applies <code>\stex_path_canonicalize:N</code> .
--	--

<code>\stex_path_to_string:NN</code> <code>\stex_path_to_string:N</code>	The inverse; turns a path into a string and stores it in the second argument variable, or 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\underline{T}</code> \star	Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment
--	---

<code>\c_stex_pwd_seq</code> <code>\c_stex_pwd_str</code> <code>\c_stex_mainfile_seq</code> <code>\c_stex_mainfile_str</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>\g_stex_currentfile_seq</code>	The file being currently processed (respecting <code>\input</code> etc.)
--------------------------------------	--

<code>\stex_filestack_push:n</code>	Push and pop (repectively) a file path to the file stack, to keep track of the current file.
<code>\stex_filestack_pop:</code>	Are called in hooks <code>file/before</code> and <code>file/after</code> , respectively.

10.1.2 MathHub Archives

<code>\mathhub</code>	We determine the path to the local MathHub folder via one of three means, in order of precedence:
<code>\c_stex_mathhub_seq</code>	
<code>\c_stex_mathhub_str</code>	

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.

<code>\l_stex_current_repository_prop</code>
--

Always points to the *current* MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the `MANIFEST.MF`-file:

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `ns`: The content namespace (for modules and symbols),
- `narr`: the narration namespace (for document references),
- `docurl`: The URL that is used as a basis for *external references*,
- `deps`: All archives that this archive depends on (currently not in use).

<code>\stex_set_current_repository:n</code>

Sets the current repository to the one with the provided ID. calls `__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

<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.
---	--

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code>
-------------------------------------	--

Change the current repository to `{<repository-name>}` (or not, if `{<repository-name>}` is empty), and passes its ID on to `{<code>}` as `#1`. Switches back to the previous repository after executing `{<code>}`.

10.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<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> <code>\mhinput</code> <hr/>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-</code> mode, inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code> <hr/>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<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(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code> <hr/>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code> <hr/>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code> <hr/>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 11

STEX-References

This sub package contains code related to links and cross-references

11.1 Macros and Environments

\STEXreftitle**\STEXreftitle{<some title>}**

Sets the title of the current document to *<some title>*. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if **\STEXreftitle{foo book}** is called, then referencing Definition 3.5 in this document in another document will display **Definition 3.5 in foo book**.

\stex_get_document_uri:

Computes the current document uri from the current archive's **narr**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URI and the reference-id.

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

Computes the current URL from the current archive's **docurl**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

11.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n**\stex_ref_new_doc_target:n{<id>}**

Sets a new reference target with id *<id>*.

\stex_ref_new_sym_target:n**\stex_ref_new_sym_target:n{<uri>}**

Sets a new reference target for the symbol *<uri>*.

11.1.2 Using References

<u><code>\sref</code></u>	<code>\sref[<i><opt-args></i>]{<i><id></i>}</code>
---------------------------	--

References the label with if *<id>*. Optional arguments: TODO

<u><code>\srefsym</code></u>	<code>\srefsym[<i><opt-args></i>]{<i><symbol></i>}</code>
------------------------------	---

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occurring in the document:

- A `\definiendum` or `\definame` for *<symbol>*,
- The `sassertion`, `sexample` or `sparagraph` with `for=<symbol>` that generated *<symbol>* in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=<symbol>`.

<u><code>\srefsymuri</code></u>	<code>\srefsymuri{<i><URI></i>}{<i><text></i>}</code>
---------------------------------	---

A convenient short-hand for `\srefsym[linktext={<text>}]<URI>`, but requires the first argument to be a full URI already. Intended to be used in e.g. `\compemph@uri`, `\defemph@uri`, etc.

Chapter 12

sTeX-Modules

This sub package contains code related to Modules

12.1 Macros and Environments

The content of a module with uri $\langle <URI> \rangle$ is stored in four macros. All modifications of these macros are global:

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

ns the *namespace* in field **ns**,

file the *file* containing the module, as a sequence of path fragments

lang the module's *language*,

sig the language of the signature module, if the current file is a translation from some other language,

deprecate if this module is deprecated, the module that replaces it,

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code> <hr/> <hr/>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
<hr/> <hr/> <code>\l_stex_all_modules_seq</code> <hr/> <hr/>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_if_in_module_p: *</code> <code>\stex_if_in_module:TF *</code> <hr/> <hr/>	Conditional for whether we are currently in a module
<hr/> <hr/> <code>\stex_if_module_exists_p:n *</code> <code>\stex_if_module_exists:nTF *</code> <hr/> <hr/>	Conditional for whether a module with the provided URI is already known.
<hr/> <hr/> <code>\stex_add_to_current_module:n</code> <code>\STEXexport</code> <hr/> <hr/>	Adds the provided tokens to the <code>_code</code> control sequence of the current module. <code>\stex_add_to_current_module:n</code> is used internally, <code>\STEXexport</code> is intended for users and additionally executes the provided code immediately.
<hr/> <hr/> <code>\stex_add_constant_to_current_module:n</code> <hr/> <hr/>	Adds the declaration with the provided name to the <code>_constants</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_add_import_to_current_module:n</code> <hr/> <hr/>	Adds the module with the provided full URI to the <code>_imports</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_collect_imports:n</code> <hr/> <hr/>	Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in <code>\l_stex_collect_imports_seq</code>
<hr/> <hr/> <code>\stex_do_up_to_module:n</code> <hr/> <hr/>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

If the current file is `.../source/sub/file.tex` in some archive with namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`. Otherwise, the namespace is the absolute file path of the current file (i.e. starting with `file:///`).

The result is stored in `\l_stex_modules_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_modules_subpath_str`.

12.1.1 The `smodule` environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`

Opens a new module with name `\langle name \rangle`. Options are:

`title` (`\langle token list \rangle`) to display in customizations.

`type` (`\langle string \rangle*`) for use in customizations.

`deprecate` (`\langle module \rangle`) if set, will throw a warning when loaded, urging to use `\langle module \rangle` instead.

`id` (`\langle string \rangle`) for cross-referencing.

`ns` (`\langle URI \rangle`) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.

`lang` (`\langle language \rangle`) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (`\langle language \rangle`) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.

`creators` (`\langle string \rangle*`) names of the creators.

`contributors` (`\langle string \rangle*`) names of contributors.

`srccite` (`\langle string \rangle`) a source citation for the content of this module.

`\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.

`\stexpatchmodule` `\stexpatchmodule [\langle type \rangle] {\langle begincode \rangle} {\langle endcode \rangle}`

Customizes the presentation for those `smodule`-environments with `type=<type>`, or all others if no `\langle type \rangle` is given.

`\STEXModule` `\STEXModule {\langle fragment \rangle}`

Attempts to find a module whose URI ends with `\langle fragment \rangle` 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.

`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists – all of which either declare module content or are needed in order to declare module content:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. sms mode continues immediately after. These macros may not take any arguments or otherwise gobble tokens.

Initially: `\makeatletter`, `\makeatother`, `\ExplSyntaxOn`, `\ExplSyntaxOff`.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed and potentially gobble up further tokens. These macros need to make sure, that the very last token they ultimately expand to is `\stex_smsmode_do:`.

Initially: `\symdecl`, `\notation`, `\symdef`, `\importmodule`, `\STEXexport`, `\inlineass`, `\inlinedef`, `\inlineex`, `\endinput`, `\setnotation`, `\copynotation`.

`\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_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

Initially: `smodule`, `copymodule`, `interpretmodule`, `sdefinition`, `sexample`, `sassertion`, `sparagraph`.

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

Tests whether SMS mode is currently active.

<code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn {<filename>} {<code>}</code>
---------------------------------------	--

Executes `<code>` in SMS mode, followed by the content of `<filename>`. `<code>` can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.

<code>\stex_smsmode_do:</code>	Starts gobbling tokens until one is encountered that is allowed in SMS mode.
--------------------------------	--

13.1.2 Imports and Inheritance

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

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

<code>\usemodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
-------------------------	---

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

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

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.

<code>\l_stex_import_name_str</code>	stores the result in these four variables.
<code>\l_stex_import_archive_str</code>	
<code>\l_stex_import_path_str</code>	
<code>\l_stex_import_ns_str</code>	

<code>\stex_import_require_module:nnnn</code>	<code>{\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}</code>
---	--

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

Finally, activates that module by executing its `_code`-macro.

Chapter 14

TeX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

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

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

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

<hr/> <hr/> <code>\stex_symdecl_do:n</code>	<p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>. Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>local</code> (boolean), • <code>type</code> (token list), • <code>args</code> (string of <code>is</code>, <code>as</code> and <code>bs</code>), • <code>arity</code> (integer string), • <code>assoc</code> (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	<p>Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.</p>
<hr/> <hr/> <code>\stex_get_symbol:n</code>	<p>Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...</p>
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[<args>]{<symbol>}{<notations⁺>}</code> Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{<URI>}{<notations⁺>}</code> Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>. Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>#<variant>#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code> (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code> Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

<hr/> <hr/>	
<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	
<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code>
	Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \SIX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	
<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
	Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \SIX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <hr/>	
<code>\stex_term_custom:nn</code>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>
	Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <hr/>	
<code>\stex_highlight_term:nn</code>	<code>\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}</code>
	Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.
<hr/> <hr/>	
<code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@uri</code>	<code>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <hr/>	
<code>\STEXinvisible</code>	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.
<hr/> <hr/>	
<code>\ellipses</code>	TODO

Chapter 16

STEX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

TeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

```
\begin{sproof}[id=simple-proof]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcases}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[type=inline] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[type=inline]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[type=inline] 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}
  \end{spfcases}
  \begin{spfstep}[type=conclusion]
    We have considered all the cases, so we have proven the assertion.
  \end{spfstep}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

18.2.4 Proof Structure

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

18.2.5 Proof End Markers

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

\sproofend	The <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i>), then this can be obtained by specifying it using the
\sProofEndSymbol	<code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code>).
Some of the proof structuring macros above will insert proof end symbols for subproofs, 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 <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> .	

18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

\pstlabelstyle	<code>\pstlabelstyle{<style>}</code> sets the style; see Figure ?? for an overview of styles. Package writers can add additional styles by adding a macro <code>\pst@make@label@<style></code> that takes
----------------	---

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

two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the \LaTeX `\@for...:=...\do{...}` macro; see Figure ?? for examples.

18.3 Limitations

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

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

21.1 Introduction

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

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

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

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

21.2 The User Interface

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

21.2.1 Package and Class Options

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

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

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

21.2.2 Document Structure

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

```
\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderiv,loadmodules]{Introducing  $\protect\bar$  Derivation
```

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.

`blindfragment`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

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

22.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

`nparagraph`
`nfragment`
`ndefinition`
`nexample`
`nsproof`
`nassertion`

22.2.3 Header and Footer Lines of the Slides

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

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

`\setsource`
`\setlicensing`

22.2.4 Frame Images

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

`\frameimage`
`\mhframeimage`

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV
Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

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

```

\stex The \TeX logo:

\sTeX

```

34 \protected\def\stex{
35   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
36 }
37 \let\sTeX\stex

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 22.)

25.3 Messages and logging

```

38 <@=stex_log>
    Warnings and error messages
39 \msg_new:nnn{stex}{error/unknownlanguage}{
40   Unknown~language:~#1
41 }
42 \msg_new:nnn{stex}{warning/nomathhub}{
43   MATHHUB~system~variable~not~found~and~no~
44   \detokenize{\mathhub}~value~set!
45 }
46 \msg_new:nnn{stex}{error/deactivated-macro}{
47   The~\detokenize{#1}~command~is~only~allowed~in~#2!
48 }

```

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

```

49 \cs_new_protected:Nn \stex_debug:nn {
50   \clist_if_in:NnTF \c_stex_debug_clist { all } {
51     \msg_set:nnn{stex}{debug / #1}{
52       \\Debug~#1:~#2\\
53     }
54     \msg_none:nn{stex}{debug / #1}
55   }{
56     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
57       \msg_set:nnn{stex}{debug / #1}{
58         \\Debug~#1:~#2\\
59       }
60       \msg_none:nn{stex}{debug / #1}
61     }
62   }
63 }

```

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

Redirecting messages:

```

64 \clist_if_in:NnTF \c_stex_debug_clist {all} {
65   \msg_redirect_module:nnn{ stex }{ none }{ term }

```

```

66 }{
67   \clist_map_inline:Nn \c_stex_debug_clist {
68     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
69   }
70 }
71
72 \stex_debug:nn{log}{debug~mode~on}

```

25.4 HTML Annotations

```

73 <@=stex_annotate>
74 \RequirePackage{rustex}

```

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

```

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

```

Conditionals for L^AT_EXML:

`\if@latexml`

```

76 \ifcsname if@latexml\endcsname\else
77   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
78 \fi

```

(End definition for `\if@latexml`. This function is documented on page 22.)

`\latexml_if_p:`

`\latexml_if:TF`

```

79 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
80   \if@latexml
81     \prg_return_true:
82   \else:
83     \prg_return_false:
84   \fi:
85 }

```

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

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

Used by annotation macros to ensure that the HTML output to annotate is not empty.

```

86 \tl_new:N \l__stex_annotate_arg_tl
87 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
88   \rustex_if:TF {
89     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
90   }{-}
91 }

```

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

`__stex_annotate_checkempty:n`

```

92 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
93   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
94   \tl_if_empty:NT \l__stex_annotate_arg_tl {
95     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
96   }
97 }

```

(End definition for `__stex_annotate_checkempty:n`.)

```

\stex_if_do_html_p: Whether to (locally) produce HTML output
\stex_if_do_html:TF
  98 \bool_new:N \_stex_html_do_output_bool
  99 \bool_set_true:N \_stex_html_do_output_bool
  100
  101 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
  102   \bool_if:nTF \_stex_html_do_output_bool
  103     \prg_return_true: \prg_return_false:
  104 }

```

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

```

\stex_suppress_html:n Whether to (locally) produce HTML output
  105 \cs_new_protected:Nn \stex_suppress_html:n {
  106   \exp_args:Nne \use:nn {
  107     \bool_set_false:N \_stex_html_do_output_bool
  108     #1
  109   }{
  110     \stex_if_do_html:T {
  111       \bool_set_true:N \_stex_html_do_output_bool
  112     }
  113   }
  114 }

```

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

`\stex_annotate:nnv` We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_ML, R_US_TE_X, p_DF_LA_TE_X).

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

`\stex_annotate_invisible:nnn`

```

  115 \rustex_if:TF{
  116   \cs_new_protected:Nn \stex_annotate:nnn {
  117     \__stex_annotate_checkempty:n { #3 }
  118     \rustex_annotate_HTML:nn {
  119       property="stex:#1" ~
  120       resource="#2"
  121     } {
  122       \mode_if_vertical:TF{
  123         \tl_use:N \l__stex_annotate_arg_tl\par
  124       }{
  125         \tl_use:N \l__stex_annotate_arg_tl
  126       }
  127     }
  128   }
  129   \cs_new_protected:Nn \stex_annotate_invisible:n {
  130     \__stex_annotate_checkempty:n { #1 }
  131     \rustex_annotate_HTML:nn {
  132       stex:visible="false" ~
  133       style:display="none"
  134     } {
  135       \mode_if_vertical:TF{
  136         \tl_use:N \l__stex_annotate_arg_tl\par
  137       }{
  138         \tl_use:N \l__stex_annotate_arg_tl
  139       }
  140     }
  141   }

```



```

140     }
141   }
142   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
143     \__stex_annotate_checkempty:n { #3 }
144     \rustex_annotate_HTML:nn {
145       property="stex:#1" ~
146       resource="#2" ~
147       stex:visible="false" ~
148       style:display="none"
149     } {
150       \mode_if_vertical:TF{
151         \tl_use:N \l__stex_annotate_arg_tl\par
152       }{
153         \tl_use:N \l__stex_annotate_arg_tl
154       }
155     }
156   }
157   \NewDocumentEnvironment{stex_annotate_env} { m m } {
158     \par
159     \rustex_annotate_HTML_begin:n {
160       property="stex:#1" ~
161       resource="#2"
162     }
163   }{
164     \par\rustex_annotate_HTML_end:
165   }
166 }{
167   \latexml_if:TF {
168     \cs_new_protected:Nn \stex_annotate:nnn {
169       \__stex_annotate_checkempty:n { #3 }
170       \mode_if_math:TF {
171         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
172           \tl_use:N \l__stex_annotate_arg_tl
173         }
174       }{
175         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
176           \tl_use:N \l__stex_annotate_arg_tl
177         }
178       }
179     }
180     \cs_new_protected:Nn \stex_annotate_invisible:n {
181       \__stex_annotate_checkempty:n { #1 }
182       \mode_if_math:TF {
183         \cs:w latexml@invisible@math\cs_end:{
184           \tl_use:N \l__stex_annotate_arg_tl
185         }
186       } {
187         \cs:w latexml@invisible@text\cs_end:{
188           \tl_use:N \l__stex_annotate_arg_tl
189         }
190       }
191     }
192     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
193       \__stex_annotate_checkempty:n { #3 }

```

```

194     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
195       \tl_use:N \l__stex_annotate_arg_tl
196     }
197   }
198   \NewDocumentEnvironment{stex_annotate_env} { m m } {
199     \par\begin{latexml@annotateenv}{#1}{#2}
200   }{
201     \par\end{latexml@annotateenv}
202   }
203 }{
204   \cs_new_protected:Nn \stex_annotate:nnn {#3}
205   \cs_new_protected:Nn \stex_annotate_invisible:n {}
206   \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
207   \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
208 }
209 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 23.)

25.5 Babel Languages

```

210 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

211 \prop_const_from_keyval:Nn \c_stex_languages_prop {
212   en = english ,
213   de = ngerman ,
214   ar = arabic ,
215   bg = bulgarian ,
216   ru = russian ,
217   fi = finnish ,
218   ro = romanian ,
219   tr = turkish ,
220   fr = french
221 }
222
223 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
224   english = en ,
225   ngerman = de ,
226   arabic = ar ,
227   bulgarian = bg ,
228   russian = ru ,
229   finnish = fi ,
230   romanian = ro ,
231   turkish = tr ,
232   french = fr
233 }
234 % todo: chinese simplified (zhs)
235 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 23.)

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

```

236 \clist_if_empty:NF \c_stex_languages_clist {
237   \clist_clear:N \l_tmpa_clist
238   \clist_map_inline:Nn \c_stex_languages_clist {
239     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
240       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
241     } {
242       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
243     }
244   }
245   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
246   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
247 }

```

25.6 Auxiliary Methods

\stex_deactivate_macro:Nn

```

248 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
249   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
250   \def#1{
251     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
252   }
253 }

```

(End definition for \stex_deactivate_macro:Nn. This function is documented on page 23.)

\stex_reactivate_macro:N

```

254 \cs_new_protected:Nn \stex_reactivate_macro:N {
255   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
256 }

```

(End definition for \stex_reactivate_macro:N. This function is documented on page 23.)

\ignorespacesandpars

```

257 \protected\def\ignorespacesandpars{
258   \begingroup\catcode13=10\relax
259   \@ifnextchar\par{
260     \endgroup\expandafter\ignorespacesandpars\@gobble
261   }{
262     \endgroup
263   }
264 }
265 \</package>

```

(End definition for \ignorespacesandpars. This function is documented on page 23.)

Chapter 26

STEX -MathHub Implementation

```
266 <*package>
267
268 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
269
270 <@@=stex_path>
271
272 Warnings and error messages
273 \msg_new:nnn{stex}{error/norepository}{
274   No~archive~#1~found~in~#2
275 }
276 \msg_new:nnn{stex}{error/notinarchive}{
277   Not~currently~in~an~archive,~but~\detokenize{#1}~
278   needs~one!
279 }
280 \msg_new:nnn{stex}{error/nofile}{
281   \detokenize{#1}~could~not~find~file~#2
282 }
283 \msg_new:nnn{stex}{error/twofiles}{
284   \detokenize{#1}~found~two~candidates~for~#2
285 }
```

26.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

```
284 \cs_new_protected:Nn \stex_path_from_string:Nn {
285   \str_set:Nx \l_tmpa_str { #2 }
286   \str_if_empty:NTF \l_tmpa_str {
287     \seq_clear:N #1
288   }{
289     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
290     \sys_if_platform_windows:T{
291       \seq_clear:N \l_tmpa_tl
```

```

292 \seq_map_inline:Nn #1 {
293   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
294   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
295 }
296 \seq_set_eq:NN #1 \l_tmpa_tl
297 }
298 \stex_path_canonicalize:N #1
299 }
300 }
301

```

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

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

```

302 \cs_new_protected:Nn \stex_path_to_string:NN {
303   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
304 }
305
306 \cs_new:Nn \stex_path_to_string:N {
307   \seq_use:Nn #1 /
308 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 24.)

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

```

309 \str_const:Nn \c__stex_path_dot_str {.}
310 \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.

```

311 \cs_new_protected:Nn \stex_path_canonicalize:N {
312   \seq_if_empty:NF #1 {
313     \seq_clear:N \l_tmpa_seq
314     \seq_get_left:NN #1 \l_tmpa_tl
315     \str_if_empty:NT \l_tmpa_tl {
316       \seq_put_right:Nn \l_tmpa_seq {}
317     }
318     \seq_map_inline:Nn #1 {
319       \str_set:Nn \l_tmpa_tl { ##1 }
320       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
321         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
322           \seq_if_empty:NNTF \l_tmpa_seq {
323             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
324               \c__stex_path_up_str
325             }
326           }{
327             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
328             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
329               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
330                 \c__stex_path_up_str
331               }
332             }{

```

```

333         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
334     }
335 }
336 }{
337     \str_if_empty:NF \l_tmpa_tl {
338         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
339     }
340 }
341 }
342 }
343 \seq_gset_eq:NN #1 \l_tmpa_seq
344 }
345 }

```

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

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

```

346 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
347     \seq_if_empty:NTF #1 {
348         \prg_return_false:
349     }{
350         \seq_get_left:NN #1 \l_tmpa_tl
351         \sys_if_platform_windows:TF{
352             \str_if_in:NnTF \l_tmpa_tl {:}{
353                 \prg_return_true:
354             }{
355                 \prg_return_false:
356             }
357         }{
358             \str_if_empty:NTF \l_tmpa_tl {
359                 \prg_return_true:
360             }{
361                 \prg_return_false:
362             }
363         }
364     }
365 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 24.)

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

366 \str_new:N\l_stex_kpsewhich_return_str
367 \cs_new_protected:Nn \stex_kpsewhich:n {
368     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
369     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
370     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
371 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

372 \sys_if_platform_windows:TF{
373   \begingroup\escapechar=-1\catcode'\=12
374   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
375   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
376   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
377   }}{
378   \stex_kpsewhich:n{-var-value~PWD}
379   }
380
381 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
382 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
383 \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 24.)

26.3 File Hooks and Tracking

```

384 <@@=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

```

385 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

386 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
387 \stex_path_from_string:Nn \c_stex_mainfile_seq
388   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 24.)

`\g_stex_currentfile_seq`

```

389 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 25.)

`\stex_filestack_push:n`

```

390 \cs_new_protected:Nn \stex_filestack_push:n {
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
392   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
393     \stex_path_from_string:Nn\g_stex_currentfile_seq{
394       \c_stex_pwd_str/#1
395     }
396   }
397   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
398   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
399 }

```

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

`\stex_filestack_pop:`

```

400 \cs_new_protected:Nn \stex_filestack_pop: {
401   \seq_if_empty:NF\g__stex_files_stack{
402     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
403   }
404   \seq_if_empty:NTF\g__stex_files_stack{
405     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
406   }{
407     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
408     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
409   }
410 }
```

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

Hooks for the current file:

```

411 \AddToHook{file/before}{
412   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
413 }
414 \AddToHook{file/after}{
415   \stex_filestack_pop:
416 }
```

26.4 MathHub Repositories

417 `<@=stex_mathhub>`

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

The path to the mathhub directory. If the `\mathhub`-macro is not set, we query `kpsewhich` for the MATHHUB system variable.

```

418 \str_if_empty:NTF\mathhub{
419   \sys_if_platform_windows:TF{
420     \begingroup\escapechar=-1\catcode'\=12
421     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
422     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
423     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
424   }{
425     \stex_kpsewhich:n{-var-value-MATHHUB}
426   }
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428 }
429 \str_if_empty:NTF\c_stex_mathhub_str{
430   \msg_warning:nn{stex}{warning/nomathhub}
431 }{
432   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
433   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434 }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
441   }
```



```

441 }
442 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 25.)

`_stex_mathhub_do_manifest:n` Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```

445 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
446   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
447     \str_set:Nx \l_tmpa_str { #1 }
448     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
449     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
450     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
451     \_stex_mathhub_find_manifest:N \l_tmpa_seq
452     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
453       \msg_error:nnxx{stex}{error/norepository}{#1}{
454         \stex_path_to_string:N \c_stex_mathhub_str
455       }
456     } {
457       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
458     }
459   }
460 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

461 \seq_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`:

```

462 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
463   \seq_set_eq:NN\l_tmpa_seq #1
464   \bool_set_true:N\l_tmpa_bool
465   \bool_while_do:Nn \l_tmpa_bool {
466     \seq_if_empty:NTF \l_tmpa_seq {
467       \bool_set_false:N\l_tmpa_bool
468     }{
469       \file_if_exist:nTF{
470         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
471       }{
472         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
473         \bool_set_false:N\l_tmpa_bool
474       }{
475         \file_if_exist:nTF{
476           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
477         }{
478           \seq_put_right:Nn\l_tmpa_seq{META-INF}
479           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

480     \bool_set_false:N\l_tmpa_bool
481   }{
482     \file_if_exist:nTF{
483       \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
484     }{
485       \seq_put_right:Nn\l_tmpa_seq{meta-inf}
486       \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
487       \bool_set_false:N\l_tmpa_bool
488     }{
489       \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
490     }
491   }
492 }
493 }
494 }
495 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
496 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

497 \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:

```

498 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
499   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
502     \str_set:Nn \l_tmpa_str {##1}
503     \exp_args:NNoo \seq_set_split:Nnn
504       \l_tmpb_seq \c_colon_str \l_tmpa_str
505     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
506       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508       }
509       \exp_args:No \str_case:nnTF \l_tmpa_tl {
510         {id} {
511           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512             { id } \l_tmpb_tl
513         }
514         {narration-base} {
515           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516             { narr } \l_tmpb_tl
517         }
518         {url-base} {
519           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520             { docurl } \l_tmpb_tl
521         }
522         {source-base} {
523           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524             { ns } \l_tmpb_tl
525         }
526       }
527     }
528 }

```

```

526     {ns} {
527         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528         { ns } \l_tmpb_tl
529     }
530     {dependencies} {
531         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532         { deps } \l_tmpb_tl
533     }
534     }{}{}
535     }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540     \stex_require_repository:n { #1 }
541     \prop_set_eq:Nc \l_stex_current_repository_prop {
542         c_stex_mathhub_#1_manifest_prop
543     }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547         \stex_debug:nn{mathhub}{Opening~archive:~#1}
548         \_stex_mathhub_do_manifest:n { #1 }
549     }
550 }

```

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

`\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 25.)

`\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     \prop_if_exist:NTF \l_stex_current_repository_prop {
572       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
573       \exp_args:Ne \l_tmpa_cs{
574         \prop_item:Nn \l_stex_current_repository_prop { id }
575     }
576   }{
577     \l_tmpa_cs{}
578   }
579 }{
580   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
581   \stex_require_repository:n \l_tmpa_str
582   \str_set:Nx \l_tmpa_str { #1 }
583   \exp_args:Nne \use:nn {
584     \stex_set_current_repository:n \l_tmpa_str
585     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
586   }{
587     \stex_debug:nn{mathhub}{switching~back~to:~
588     \prop_if_exist:NTF \l_stex_current_repository_prop {
589       \prop_item:Nn \l_stex_current_repository_prop { id }::~
590     \meaning\l_stex_current_repository_prop
591   }{
592     no~repository
593   }
594 }
595 \prop_if_exist:NTF \l_stex_current_repository_prop {
596   \stex_set_current_repository:n {
597     \prop_item:Nn \l_stex_current_repository_prop { id }
598   }
599 }{
600   \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
601 }
602 }
603 }
604 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [25](#).)

26.5 Using Content in Archives

`\mhpath`

```

605 \def \mhpath #1 #2 {
606   \exp_args:Ne \tl_if_empty:nTF{#1}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610 }{
611   \c_stex_mathhub_str / #1 / source / #2

```

```

612 }
613 }

```

(End definition for `\mhpath`. This function is documented on page 26.)

`\inputref`
`\mhinput`

```

614 \newif \ifinputref \inputreffalse
615
616 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617   \stex_in_repository:nn {#1} {
618     \ifinputref
619       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620     \else
621       \inputreftrue
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \inputreffalse
624   \fi
625 }
626 }
627 \NewDocumentCommand \mhinput { 0{} m}{
628   \stex_mhinput:nn{ #1 }{ #2 }
629 }
630
631 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632   \stex_in_repository:nn {#1} {
633     \bool_lazy_any:nTF {
634       {\rustex_if_p:}
635       {\latexml_if_p:}
636     } {
637       \str_clear:N \l_tmpa_str
638       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640       }
641       \stex_annotate_invisible:nnn{inputref}{
642         \l_tmpa_str / #2
643       }{}
644     }{
645       \begingroup
646         \inputreftrue
647         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
648       \endgroup
649     }
650   }
651 }
652 \NewDocumentCommand \inputref { 0{} m}{
653   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
654 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 26.)

`\addmhbibresource`

```

655 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }

```

```

659 }
660 \newcommand\addmhbibresource[2][]{
661   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
662 }

```

(End definition for \addmhbibresource. This function is documented on page 26.)

\libinput

```

663 \cs_new_protected:Npn \libinput #1 {
664   \prop_if_exist:NF \l_stex_current_repository_prop {
665     \msg_error:nnn{stex}{error/notinarchive}\libinput
666   }
667   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
668     \msg_error:nnn{stex}{error/notinarchive}\libinput
669   }
670   \seq_clear:N \l__stex_mathhub_libinput_files_seq
671   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
672   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
673
674   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
675     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
676     \IfFileExists{ \l_tmpa_str }{
677       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
678     }{}
679     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
680     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
681   }
682
683   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
684   \IfFileExists{ \l_tmpa_str }{
685     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
686   }{}
687
688   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
689     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
690   }{
691     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
692       \input{ ##1 }
693     }
694   }
695 }

```

(End definition for \libinput. This function is documented on page 26.)

\libusepackage

```

696 \NewDocumentCommand \libusepackage {0{} m} {
697   \prop_if_exist:NF \l_stex_current_repository_prop {
698     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
699   }
700   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
701     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
702   }
703   \seq_clear:N \l__stex_mathhub_libinput_files_seq
704   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
705   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

706
707 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
708   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
709   \IfFileExists{ \l_tmpa_str.sty }{
710     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
711   }{}
712   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
713   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
714 }
715
716 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
717 \IfFileExists{ \l_tmpa_str.sty }{
718   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
719 }{}
720
721 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
722   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
723 }{
724   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
725     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
726       \usepackage[#1]{ #1 }
727     }
728   }{
729     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
730   }
731 }
732 }

```

(End definition for `\libusepackage`. This function is documented on page 26.)

`\mhgraphics`
`\cmhgraphics`

```

733
734 \AddToHook{begindocument}{
735   \ltx@ifpackageloaded{graphicx}{
736     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
737     \newcommand\mhgraphics[2][]{\%
738       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
739       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
740     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
741   }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 26.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

742 \ltx@ifpackageloaded{listings}{
743   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
744   \newcommand\lstinputmhlisting[2][]{\%
745     \def\lst@mhrepos{}\setkeys{lst}{#1}%
746     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
747   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
748 }{}
749 }
750
751 </package>

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 26.)

Chapter 27

STEX -References Implementation

```
752 <*package>
753
754 %%%%%%%%%%%%%% references.dtx %%%%%%%%%%%%%%
755
756 <@@=stex_refs>
    Warnings and error messages
757
```

References are stored in the file `\jobname.sref`, to enable cross-referencing external documents.

```
758 %\iow_new:N \c__stex_refs_refs_iow
759 \AddToHook{begindocument}{
760 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
761 }
762 \AddToHook{enddocument}{
763 % \iow_close:N \c__stex_refs_refs_iow
764 }
```

`\STEXreftitle`

```
765 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
766
767 \NewDocumentCommand \STEXreftitle { m } {
768 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
769 }
```

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

27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
770 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 27.)

`\stex_get_document_uri:`

```
771 \cs_new_protected:Nn \stex_get_document_uri: {  
772   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
773   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
774   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
775   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
776   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
777  
778   \str_clear:N \l_tmpa_str  
779   \prop_if_exist:NT \l_stex_current_repository_prop {  
780     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
781       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
782     }  
783   }  
784  
785   \str_if_empty:NTF \l_tmpa_str {  
786     \str_set:Nx \l_stex_current_docns_str {  
787       file:/\stex_path_to_string:N \l_tmpa_seq  
788     }  
789   }{  
790     \bool_set_true:N \l_tmpa_bool  
791     \bool_while_do:Nn \l_tmpa_bool {  
792       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
793       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
794         {source} { \bool_set_false:N \l_tmpa_bool }  
795       }{}{  
796         \seq_if_empty:NT \l_tmpa_seq {  
797           \bool_set_false:N \l_tmpa_bool  
798         }  
799       }  
800     }  
801  
802     \seq_if_empty:NTF \l_tmpa_seq {  
803       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
804     }{  
805       \str_set:Nx \l_stex_current_docns_str {  
806         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
807       }  
808     }  
809   }  
810 }
```

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

`\l_stex_current_docurl_str`

```
811 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 27.)

`\stex_get_document_url:`

```
812 \cs_new_protected:Nn \stex_get_document_url: {  
813   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
814   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
815   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

816 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
817 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
818
819 \str_clear:N \l_tmpa_str
820 \prop_if_exist:NT \l_stex_current_repository_prop {
821   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
822     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824     }
825   }
826 }
827
828 \str_if_empty:NTF \l_tmpa_str {
829   \str_set:Nx \l_stex_current_docurl_str {
830     file:/\stex_path_to_string:N \l_tmpa_seq
831   }
832 }{
833   \bool_set_true:N \l_tmpa_bool
834   \bool_while_do:Nn \l_tmpa_bool {
835     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
836     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
837       {source} { \bool_set_false:N \l_tmpa_bool }
838     }{}{
839       \seq_if_empty:NT \l_tmpa_seq {
840         \bool_set_false:N \l_tmpa_bool
841       }
842     }
843   }
844
845   \seq_if_empty:NTF \l_tmpa_seq {
846     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
847   }{
848     \str_set:Nx \l_stex_current_docurl_str {
849       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
850     }
851   }
852 }
853 }

```

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

27.2 Setting Reference Targets

```

854 \str_const:Nn \c__stex_refs_url_str{URL}
855 \str_const:Nn \c__stex_refs_ref_str{REF}
856 \str_new:N \l__stex_refs_curr_label_str
857 % @currentlabel -> number
858 % @currentlabelname -> title
859 % @currentHref -> name.number <- id of some kind
860 % \theH# -> \arabic{section}
861 % \the# -> number
862 % \hyper@makecurrent{#}
863 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

864 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
865   \stex_get_document_uri:
866   \str_clear:N \l__stex_refs_curr_label_str
867   \str_set:Nx \l_tmpa_str { #1 }
868   \str_if_empty:NT \l_tmpa_str {
869     \int_incr:N \l__stex_refs_unnamed_counter_int
870     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
871   }
872   \str_set:Nx \l__stex_refs_curr_label_str {
873     \l_stex_current_docns_str?\l_tmpa_str
874   }
875   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
876     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
877   }
878   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
879     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
880   }
881   \stex_if_smsmode:TF {
882     \stex_get_document_url:
883     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
884     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
885   }{
886     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
887     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
888     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
889     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
890   }
891 }

```

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

The following is used to set the necessary macros in the .aux-file.

```

892 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
893   \str_set:Nn \l_tmpa_str {#1?#2}
894   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
895   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
896     \seq_new:c {g__stex_refs_labels_#2_seq}
897   }
898   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
899     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
900   }
901 }

```

To avoid resetting the same macros when the .aux-file is read at the end of the document:

```

902 \AtEndDocument{
903   \def\stexauxadddocref#1 #2 {}{}
904 }

```

`\stex_ref_new_sym_target:n`

```

905 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
906   \stex_if_smsmode:TF {
907     \str_if_exist:cF{sref_sym_#1_type}{
908       \stex_get_document_url:
909       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

910     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
911   }
912 }{
913   \str_if_empty:NF \l__stex_refs_curr_label_str {
914     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
915     \immediate\write\@auxout{
916       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
917         \l__stex_refs_curr_label_str
918     }
919   }
920 }
921 }
922 }

```

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

27.3 Using References

```

923 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

924
925 \keys_define:nn { stex / sref } {
926   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
927   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
928   pre           .tl_set:N = \l__stex_refs_pre_tl ,
929   post          .tl_set:N = \l__stex_refs_post_tl ,
930 }
931 \cs_new_protected:Nn \__stex_refs_args:n {
932   \tl_clear:N \l__stex_refs_linktext_tl
933   \tl_clear:N \l__stex_refs_fallback_tl
934   \tl_clear:N \l__stex_refs_pre_tl
935   \tl_clear:N \l__stex_refs_post_tl
936   \str_clear:N \l__stex_refs_repo_str
937   \keys_set:nn { stex / sref } { #1 }
938 }

```

The actual macro:

```

939 \NewDocumentCommand \sref { 0{} m}{
940   \__stex_refs_args:n { #1 }
941   \str_if_empty:NTF \l__stex_refs_indocument_str {
942     \str_set:Nx \l_tmpa_str { #2 }
943     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
944     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
945       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
946         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
947           \str_clear:N \l_tmpa_str
948         }
949       }{
950         \str_clear:N \l_tmpa_str
951       }
952     }{
953       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
954       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

955 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
956 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
957   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
958   \str_clear:N \l_tmpa_str
959   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
960     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
961       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
962     }{
963       \seq_map_break:n {
964         \str_set:Nn \l_tmpa_str { ##1 }
965       }
966     }
967   }
968 }{
969   \str_clear:N \l_tmpa_str
970 }
971 }
972 \str_if_empty:NTF \l_tmpa_str {
973   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
974 }{
975   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
976     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
977       \cs_if_exist:cTF{autoref}{
978         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
979       }{
980         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
981       }
982     }{
983       \ltx@ifpackageloaded{hyperref}{
984         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
985       }{
986         \l__stex_refs_linktext_tl
987       }
988     }
989   }{
990     \ltx@ifpackageloaded{hyperref}{
991       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
992     }{
993       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
994     }
995   }
996 }
997 }{
998   % TODO
999 }
1000 }

```

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

`\srefsym`

```

1001 \NewDocumentCommand \srefsym { 0{} m}{
1002   \stex_get_symbol:n { #2 }
1003   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1004 }

```

```

1005
1006 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1007   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1008     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1009   }{
1010     \__stex_refs_args:n { #1 }
1011     \str_if_empty:NTF \l__stex_refs_indocument_str {
1012       \tl_if_exist:cTF{sref_sym_#2 _type}{
1013         % doc uri in \l_tmpb_str
1014         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1015         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1016           % reference
1017           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1018             \cs_if_exist:cTF{autoref}{
1019               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1020             }{
1021               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1022             }
1023           }{
1024             \ltx@ifpackageloaded{hyperref}{
1025               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1026             }{
1027               \l__stex_refs_linktext_tl
1028             }
1029           }
1030         }{
1031           % URL
1032           \ltx@ifpackageloaded{hyperref}{
1033             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1034           }{
1035             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1036           }
1037         }
1038       }{
1039         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1040       }
1041     }{
1042       % TODO
1043     }
1044   }
1045 }

```

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

\srefsymuri

```

1046 \cs_new_protected:Npn \srefsymuri #1 #2 {
1047   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1048 }

```

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

```

1049 </package>

```

Chapter 28

STEX -Modules Implementation

```
1050 <*package>
1051
1052 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1053
1054 <@@=stex_modules>
1055
1056   Warnings and error messages
1057   \msg_new:nnn{stex}{error/unknownmodule}{
1058     No~module~#1~found
1059   }
1060   \msg_new:nnn{stex}{error/syntax}{
1061     Syntax~error:~#1
1062   }
1063   \msg_new:nnn{stex}{error/siglanguage}{
1064     Module~#1~declares~signature~#2,~but~does~not~
1065     declare~its~language
1066   }
1067   \msg_new:nnn{stex}{warning/deprecated}{
1068     #1~is~deprecated;~please~use~#2~instead!
1069   }
1070   \msg_new:nnn{stex}{error/conflictingmodules}{
1071     Conflicting~imports~for~module~#1
1072   }
1073
1074 \l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
1077 (End definition for \l_stex_current_module_str. This variable is documented on page 30.)
1078
1079 \l_stex_all_modules_seq Stores all available modules
1080 \seq_new:N \l_stex_all_modules_seq
1081
1082 (End definition for \l_stex_all_modules_seq. This variable is documented on page 30.)
```



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1074 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1075   \str_if_empty:NTF \l_stex_current_module_str
1076   \prg_return_false: \prg_return_true:
1077 }

```

(End definition for \stex_if_in_module:TF. This function is documented on page 30.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1078 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1079   \prop_if_exist:cTF { c_stex_module_#1_prop }
1080   \prg_return_true: \prg_return_false:
1081 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 30.)

\stex_add_to_current_module:n Only allowed within modules:

```

\STEXexport
1082 \cs_new_protected:Nn \stex_add_to_current_module:n {
1083   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1084 }
1085 \cs_new_protected:Npn \STEXexport {
1086   \begingroup
1087   \newlinechar=-1\relax
1088   \endlinechar=-1\relax
1089   %\catcode'\ = 9\relax
1090   \expandafter\endgroup\__stex_modules_export:n
1091 }
1092 \cs_new_protected:Nn \__stex_modules_export:n {
1093   \ignorespaces #1
1094   \stex_add_to_current_module:n { \ignorespaces #1 }
1095   \stex_smsmode_do:
1096 }
1097 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 30.)

```

\stex_add_constant_to_current_module:n
1098 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1099   \str_set:Nx \l_tmpa_str { #1 }
1100   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1101 }

```

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

```

\stex_add_import_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \exp_args:Nno
1105   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1106     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1107   }
1108 }

```

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

`\stex_collect_imports:n`

```

1109 \cs_new_protected:Nn \stex_collect_imports:n {
1110   \seq_clear:N \l_stex_collect_imports_seq
1111   \__stex_modules_collect_imports:n {#1}
1112 }
1113 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1114   \seq_map_inline:cn {c_stex_module_#1_imports} {
1115     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1116       \__stex_modules_collect_imports:n { ##1 }
1117     }
1118   }
1119   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1120     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1121   }
1122 }

```

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

`\stex_do_up_to_module:n`

```

1123 \int_new:N \l__stex_modules_group_depth_int
1124 \tl_new:N \l__stex_modules_aftergroup_tl
1125 \cs_new_protected:Nn \stex_do_up_to_module:n {
1126   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1127     #1
1128   }{
1129     #1
1130     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1131       \aftergroup\__stex_modules_aftergroup_do:
1132     }
1133   }
1134   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1135     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1136       \l__stex_modules_aftergroup_tl
1137       \tl_clear:N \l__stex_modules_aftergroup_tl
1138     }{
1139       \l__stex_modules_aftergroup_tl
1140       \aftergroup\__stex_modules_aftergroup_do:
1141     }
1142   }

```

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

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

1143

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

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```

1144 \str_new:N \l_stex_modules_ns_str
1145 \str_new:N \l_stex_modules_subpath_str

```

```

1146 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1147   \str_set:Nx \l_tmpa_str { #1 }
1148   \seq_set_eq:NN \l_tmpa_seq #2
1149   % split off file extension
1150   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1151   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1152   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1153   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1154
1155   \bool_set_true:N \l_tmpa_bool
1156   \bool_while_do:Nn \l_tmpa_bool {
1157     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1158     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1159       {source} { \bool_set_false:N \l_tmpa_bool }
1160     }{}{
1161       \seq_if_empty:NT \l_tmpa_seq {
1162         \bool_set_false:N \l_tmpa_bool
1163       }
1164     }
1165   }
1166
1167   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1168   \str_if_empty:NTF \l_stex_modules_subpath_str {
1169     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1170   }{
1171     \str_set:Nx \l_stex_modules_ns_str {
1172       \l_tmpa_str/\l_stex_modules_subpath_str
1173     }
1174   }
1175 }
1176
1177 \cs_new_protected:Nn \stex_modules_current_namespace: {
1178   \str_clear:N \l_stex_modules_subpath_str
1179   \prop_if_exist:NTF \l_stex_current_repository_prop {
1180     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1181     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1182   }{
1183     % split off file extension
1184     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1186     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1187     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1188     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1189     \str_set:Nx \l_stex_modules_ns_str {
1190       file:\stex_path_to_string:N \l_tmpa_seq
1191     }
1192   }
1193 }

```

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

28.1 The smodule environment

smodule arguments:

```

1194 \keys_define:nn { stex / module } {
1195   title      .tl_set:N      = \smoduletitle ,
1196   type       .str_set_x:N   = \smoduletype ,
1197   id         .str_set_x:N   = \smoduleid ,
1198   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1199   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1200   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1201   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1202   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1203   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1204   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1205   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1206 }
1207
1208 \cs_new_protected:Nn \__stex_modules_args:n {
1209   \str_clear:N \smoduletitle
1210   \str_clear:N \smoduletype
1211   \str_clear:N \smoduleid
1212   \str_clear:N \l_stex_module_ns_str
1213   \str_clear:N \l_stex_module_deprecate_str
1214   \str_clear:N \l_stex_module_lang_str
1215   \str_clear:N \l_stex_module_sig_str
1216   \str_clear:N \l_stex_module_creators_str
1217   \str_clear:N \l_stex_module_contributors_str
1218   \str_clear:N \l_stex_module_meta_str
1219   \str_clear:N \l_stex_module_srccite_str
1220   \keys_set:nn { stex / module } { #1 }
1221 }
1222
1223 % module parameters here? In the body?
1224
```

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

```

1225 \cs_new_protected:Nn \stex_module_setup:nn {
1226   \str_set:Nx \l_stex_module_name_str { #2 }
1227   \__stex_modules_args:n { #1 }

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

1228   \stex_if_in_module:TF {
1229     % Nested module
1230     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1231       { ns } \l_stex_module_ns_str
1232     \str_set:Nx \l_stex_module_name_str {
1233       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1234       { name } / \l_stex_module_name_str
1235     }
1236   }{
1237     % not nested:
1238     \str_if_empty:NT \l_stex_module_ns_str {
1239       \stex_modules_current_namespace:

```

```

1240 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1241 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1242 / {\l_stex_module_ns_str}
1243 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1244 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1245 \str_set:Nx \l_stex_module_ns_str {
1246 \stex_path_to_string:N \l_tmpa_seq
1247 }
1248 }
1249 }
1250 }

```

Next, we determine the language of the module:

```

1251 \str_if_empty:NT \l_stex_module_lang_str {
1252 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1253 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1254 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1255 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1256 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1257 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1258 inferred~from~file~name}
1259 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1260 }
1261 }
1262
1263 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1264 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1265 \l_tmpa_str {
1266 \ltx@ifpackageloaded{babel}{
1267 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1268 }{}
1269 } {
1270 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1271 }
1272 }}

```

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

```

1273 \str_if_empty:NTF \l_stex_module_sig_str {
1274 \exp_args:Nnx \prop_gset_from_keyval:cn {
1275 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1276 } {
1277 name = \l_stex_module_name_str ,
1278 ns = \l_stex_module_ns_str ,
1279 file = \exp_not:o { \g_stex_currentfile_seq } ,
1280 lang = \l_stex_module_lang_str ,
1281 sig = \l_stex_module_sig_str ,
1282 deprecate = \l_stex_module_deprecate_str ,
1283 meta = \l_stex_module_meta_str
1284 }
1285 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1286 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1287 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1288 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1289 \str_if_empty:NT \l_stex_module_meta_str {
1290   \str_set:Nx \l_stex_module_meta_str {
1291     \c_stex_metatheory_ns_str ? Metatheory
1292   }
1293 }
1294 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1295   \bool_set_true:N \l_stex_in_meta_bool
1296   \exp_args:Nx \stex_add_to_current_module:n {
1297     \bool_set_true:N \l_stex_in_meta_bool
1298     \stex_activate_module:n {\l_stex_module_meta_str}
1299     \bool_set_false:N \l_stex_in_meta_bool
1300   }
1301   \stex_activate_module:n {\l_stex_module_meta_str}
1302   \bool_set_false:N \l_stex_in_meta_bool
1303 }
1304 }{
1305   \str_if_empty:NT \l_stex_module_lang_str {
1306     \msg_error:nnxx{stex}{error/siglanguage}{
1307       \l_stex_module_ns_str?\l_stex_module_name_str
1308     }{\l_stex_module_sig_str}
1309   }
1310
1311   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1312   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1313   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1314   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1315   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1316   \str_set:Nx \l_tmpa_str {
1317     \stex_path_to_string:N \l_tmpa_seq /
1318     \l_tmpa_str . \l_stex_module_sig_str .tex
1319   }
1320   \IfFileExists \l_tmpa_str {
1321     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1322       \str_clear:N \l_stex_current_module_str
1323       \seq_clear:N \l_stex_all_modules_seq
1324       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1325     }
1326   }{
1327     \msg_error:nnxx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1328   }
1329   \stex_if_smsmode:F {
1330     \stex_activate_module:n {
1331       \l_stex_module_ns_str ? \l_stex_module_name_str
1332     }
1333   }
1334   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1335 }
1336 \str_if_empty:NF \l_stex_module_deprecate_str {
1337   \msg_warning:nnxx{stex}{warning/deprecated}{
1338     Module~\l_stex_current_module_str
1339   }{
1340     \l_stex_module_deprecate_str
1341   }

```

```

1342 }
1343 \seq_put_right:Nx \l_stex_all_modules_seq {
1344   \l_stex_module_ns_str ? \l_stex_module_name_str
1345 }
1346 }

```

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

smodule The module environment.

`_stex_modules_begin_module:` implements `\begin{smodule}`

```

1347 \cs_new_protected:Nn \_stex_modules_begin_module: {
1348   \stex_reactivate_macro:N \STEXexport
1349   \stex_reactivate_macro:N \importmodule
1350   \stex_reactivate_macro:N \symdecl
1351   \stex_reactivate_macro:N \notation
1352   \stex_reactivate_macro:N \symdef
1353
1354   \stex_debug:nn{modules}{
1355     New~module:\\
1356     Namespace:~\l_stex_module_ns_str\\
1357     Name:~\l_stex_module_name_str\\
1358     Language:~\l_stex_module_lang_str\\
1359     Signature:~\l_stex_module_sig_str\\
1360     Metatheory:~\l_stex_module_meta_str\\
1361     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1362   }
1363
1364   \stex_if_smsmode:F{
1365     \begin{stex_annotate_env} {theory} {
1366       \l_stex_module_ns_str ? \l_stex_module_name_str
1367     }
1368
1369     \stex_annotate_invisible:nnn{header}{} {
1370       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1371       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1372       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1373         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1374       }
1375       \str_if_empty:NF \smoduletype {
1376         \stex_annotate:nnn{type}{\smoduletype}{}
1377       }
1378     }
1379   }
1380   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1381   % TODO: Inherit metatheory for nested modules?
1382 }
1383 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:.`)

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

```

1384 \cs_new_protected:Nn \_stex_modules_end_module: {
1385   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1386 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1387 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1388 \NewDocumentEnvironment { smodule } { 0{} m } {
1389   \stex_module_setup:nn{#1}{#2}
1390   \par
1391   \stex_if_smsmode:F{
1392     \tl_clear:N \l_tmpa_tl
1393     \clist_map_inline:Nn \smoduletype {
1394       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1395         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1396       }
1397     }
1398     \tl_if_empty:NTF \l_tmpa_tl {
1399       \__stex_modules_smodule_start:
1400     }{
1401       \l_tmpa_tl
1402     }
1403   }
1404   \__stex_modules_begin_module:
1405   \str_if_empty:NF \smoduleid {
1406     \stex_ref_new_doc_target:n \smoduleid
1407   }
1408   \stex_smsmode_do:
1409 } {
1410   \__stex_modules_end_module:
1411   \stex_if_smsmode:F {
1412     \end{stex_annotate_env}
1413     \clist_set:Nn \l_tmpa_clist \smoduletype
1414     \tl_clear:N \l_tmpa_tl
1415     \clist_map_inline:Nn \l_tmpa_clist {
1416       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1417         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1418       }
1419     }
1420     \tl_if_empty:NTF \l_tmpa_tl {
1421       \__stex_modules_smodule_end:
1422     }{
1423       \l_tmpa_tl
1424     }
1425   }
1426 }

```

\stexpatchmodule

```

1427 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1428 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1429
1430 \newcommand\stexpatchmodule[3] [] {
1431   \str_set:Nx \l_tmpa_str{ #1 }
1432   \str_if_empty:NTF \l_tmpa_str {
1433     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1434     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1435   }{

```



```

1436     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1437     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1438   }
1439 }

```

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

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1440 \NewDocumentCommand \STEXModule { m } {
1441   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1442   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1443   \tl_set:Nn \l_tmpa_tl {
1444     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1445   }
1446   \seq_map_inline:Nn \l_stex_all_modules_seq {
1447     \str_set:Nn \l_tmpb_str { ##1 }
1448     \str_if_eq:eeT { \l_tmpa_str } {
1449       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1450     } {
1451       \seq_map_break:n {
1452         \tl_set:Nn \l_tmpa_tl {
1453           \stex_invoke_module:n { ##1 }
1454         }
1455       }
1456     }
1457   }
1458   \l_tmpa_tl
1459 }
1460
1461 \cs_new_protected:Nn \stex_invoke_module:n {
1462   \stex_debug:nn{modules}{Invoking~module~#1}
1463   \peek_charcode_remove:NTF ! {
1464     \__stex_modules_invoke_uri:nN { #1 }
1465   } {
1466     \peek_charcode_remove:NTF ? {
1467       \__stex_modules_invoke_symbol:nn { #1 }
1468     } {
1469       \msg_error:nnx{stex}{error/syntax}{
1470         ?~or~!~expected~after~
1471         \c_backslash_str STEXModule{#1}
1472       }
1473     }
1474   }
1475 }
1476
1477 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1478   \str_set:Nn #2 { #1 }
1479 }
1480
1481 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1482   \stex_invoke_symbol:n{#1?#2}

```

```
1483 }
```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 31.)

`\stex_activate_module:n`

```
1484 \bool_new:N \l_stex_in_meta_bool
1485 \bool_set_false:N \l_stex_in_meta_bool
1486 \cs_new_protected:Nn \stex_activate_module:n {
1487   \stex_debug:nn{modules}{Activating~module~#1}
1488   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1489     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1490   }
1491   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1492     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1493     \use:c{ c_stex_module_#1_code }
1494   }
1495 }
```

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

```
1496 \</package>
```

Chapter 29

STEX -Module Inheritance Implementation

```
1497 <*package>
1498
1499 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1500
```

29.1 SMS Mode

```
1501 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1502 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1503 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1504 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1505
1506 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1507   \makeatletter
1508   \makeatother
1509   \ExplSyntaxOn
1510   \ExplSyntaxOff
1511   \rustexBREAK
1512 }
1513
1514 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1515   \symdef
1516   \importmodule
1517   \notation
1518   \symdecl
1519   \STEXexport
1520   \inlineass
1521   \inlinedef
1522   \inlineex
1523   \endinput
1524   \setnotation
```

```

1525 \copynotation
1526 }
1527
1528 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1529   \tl_to_str:n {
1530     smodule,
1531     copymodule,
1532     interpretmodule,
1533     sdefinition,
1534     sexample,
1535     sassertion,
1536     sparagraph
1537   }
1538 }

```

(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 33.)

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```

1539 \bool_new:N \g__stex_smsmode_bool
1540 \bool_set_false:N \g__stex_smsmode_bool
1541 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1542   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1543 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1544 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1545   \vbox_set:Nn \l_tmpa_box {
1546     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1547     \bool_gset_true:N \g__stex_smsmode_bool
1548     #2
1549     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1550   }
1551   \box_clear:N \l_tmpa_box
1552 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1553 \quark_new:N \q__stex_smsmode_break
1554
1555 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1556   \stex_filestack_push:n{#1}
1557   \_stex_smsmode_in_smsmode:nn{#1} {
1558     #2
1559     \everyeof{\q__stex_smsmode_break\noexpand}
1560     \expandafter\expandafter\expandafter
1561     \stex_smsmode_do:
1562     \csname @ @ input\endcsname "#1"\relax
1563   }
1564   \stex_filestack_pop:
1565 }

```

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

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

```

1566 \cs_new_protected:Npn \stex_smsmode_do: {
1567   \stex_if_smsmode:T {
1568     \__stex_smsmode_do:w
1569   }
1570 }
1571 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1572   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1573     \expandafter\if\expandafter\relax\noexpand#1
1574     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1575   } \else\expandafter\__stex_smsmode_do:w\fi
1576 }{
1577   \__stex_smsmode_do:w % #1
1578 }
1579 }
1580 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1581   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1582     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1583       #1\__stex_smsmode_do:w
1584     }{
1585       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1586         #1
1587       }{
1588         \cs_if_eq:NNTF \begin #1 {
1589           \__stex_smsmode_check_begin:n
1590         }{
1591           \cs_if_eq:NNTF \end #1 {
1592             \__stex_smsmode_check_end:n
1593           }{
1594             \__stex_smsmode_do:w
1595           }
1596         }
1597       }
1598     }
1599   }
1600 }
1601
1602 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1603   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1604     \begin{#1}
1605   }{
1606     \__stex_smsmode_do:w
1607   }
1608 }
1609 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1610   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1611     \end{#1}\__stex_smsmode_do:w
1612   }{
1613     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1614   }
1615 }
```

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

29.2 Inheritance

```

1616 <@@=stex_importmodule>

\stex_import_module_uri:nn

1617 \cs_new_protected:Nn \stex_import_module_uri:nn {
1618   \str_set:Nx \l_stex_import_archive_str { #1 }
1619   \str_set:Nn \l_stex_import_path_str { #2 }
1620
1621   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1622   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1623   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1624
1625   \stex_modules_current_namespace:
1626   \bool_lazy_all:nTF {
1627     {\str_if_empty_p:N \l_stex_import_archive_str}
1628     {\str_if_empty_p:N \l_stex_import_path_str}
1629     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1630   }{
1631     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1632     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1633   }{
1634     \str_if_empty:NT \l_stex_import_archive_str {
1635       \prop_if_exist:NT \l_stex_current_repository_prop {
1636         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1637       }
1638     }
1639     \str_if_empty:NTF \l_stex_import_archive_str {
1640       \str_if_empty:NF \l_stex_import_path_str {
1641         \str_set:Nx \l_stex_import_ns_str {
1642           \l_stex_module_ns_str / \l_stex_import_path_str
1643         }
1644       }
1645     }{
1646       \stex_require_repository:n \l_stex_import_archive_str
1647       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1648       \l_stex_import_ns_str
1649       \str_if_empty:NF \l_stex_import_path_str {
1650         \str_set:Nx \l_stex_import_ns_str {
1651           \l_stex_import_ns_str / \l_stex_import_path_str
1652         }
1653       }
1654     }
1655   }
1656 }
```

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

<code>\l_stex_import_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_import_archive_str</code>	1657 \str_new:N \l_stex_import_name_str
<code>\l_stex_import_path_str</code>	1658 \str_new:N \l_stex_import_archive_str
<code>\l_stex_import_ns_str</code>	1659 \str_new:N \l_stex_import_path_str

```
1660 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 35.)

```
\stex_import_require_module:nnnn
    {\ns} {\{archive-ID\}} {\{path\}} {\{name\}}
1661 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1662   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1663
1664     % archive
1665     \str_set:Nx \l_tmpa_str { #2 }
1666     \str_if_empty:NTF \l_tmpa_str {
1667       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1668     } {
1669       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1670       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1671       \seq_put_right:Nn \l_tmpa_seq { source }
1672     }
1673
1674     % path
1675     \str_set:Nx \l_tmpb_str { #3 }
1676     \str_if_empty:NTF \l_tmpb_str {
1677       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1678
1679       \ltx@ifpackageloaded{babel} {
1680         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1681           { \language } \l_tmpb_str {
1682           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1683         }
1684       } {
1685         \str_clear:N \l_tmpb_str
1686       }
1687
1688       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1689       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1690         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1691       }{
1692         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1693         \IfFileExists{ \l_tmpa_str.tex }{
1694           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1695         }{
1696           % try english as default
1697           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1698           \IfFileExists{ \l_tmpa_str.en.tex }{
1699             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1700           }{
1701             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1702           }
1703         }
1704       }
1705
1706     } {
1707       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1708       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1709
```

```

1710 \ltx@ifpackageloaded{babel} {
1711   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1712     { \language } \l_tmpb_str {
1713       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1714     }
1715   } {
1716     \str_clear:N \l_tmpb_str
1717   }
1718
1719   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1720
1721   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1722   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1723     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1724   }{
1725     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1726     \IfFileExists{ \l_tmpa_str/#4.tex }{
1727       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1728     }{
1729       % try english as default
1730       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1731       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1732         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1733       }{
1734         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1735         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1736           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1737         }{
1738           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1739           \IfFileExists{ \l_tmpa_str.tex }{
1740             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1741           }{
1742             % try english as default
1743             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1744             \IfFileExists{ \l_tmpa_str.en.tex }{
1745               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1746             }{
1747               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1748             }
1749           }
1750         }
1751       }
1752     }
1753   }
1754 }
1755
1756 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1757   \seq_clear:N \l_stex_all_modules_seq
1758   \str_clear:N \l_stex_current_module_str
1759   \str_set:Nx \l_tmpb_str { #2 }
1760   \str_if_empty:NF \l_tmpb_str {
1761     \stex_set_current_repository:n { #2 }
1762   }
1763   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```



```

1764     }
1765
1766     \stex_if_module_exists:nF { #1 ? #4 } {
1767         \msg_error:nnx{stex}{error/unknownmodule}{
1768             #1?#4~(in~file~\g__stex_importmodule_file_str)
1769         }
1770     }
1771 }
1772 \stex_activate_module:n { #1 ? #4 }
1773 }

```

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

`\importmodule`

```

1774 \NewDocumentCommand \importmodule { 0{} m } {
1775     \stex_import_module_uri:nn { #1 } { #2 }
1776     \stex_debug:nn{modules}{Importing~module:~
1777         \l_stex_import_ns_str ? \l_stex_import_name_str
1778     }
1779     \stex_if_smsmode:F {
1780         \stex_import_require_module:nnnn
1781         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1782         { \l_stex_import_path_str } { \l_stex_import_name_str }
1783         \stex_annotate_invisible:nnn
1784         {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1785     }
1786     \exp_args:Nx \stex_add_to_current_module:n {
1787         \stex_import_require_module:nnnn
1788         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1789         { \l_stex_import_path_str } { \l_stex_import_name_str }
1790     }
1791     \exp_args:Nx \stex_add_import_to_current_module:n {
1792         \l_stex_import_ns_str ? \l_stex_import_name_str
1793     }
1794     \stex_smsmode_do:
1795     \ignorespacesandpars
1796 }
1797 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

1798 \NewDocumentCommand \usemodule { 0{} m } {
1799     \stex_if_smsmode:F {
1800         \stex_import_module_uri:nn { #1 } { #2 }
1801         \stex_import_require_module:nnnn
1802         { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1803         { \l_stex_import_path_str } { \l_stex_import_name_str }
1804         \stex_annotate_invisible:nnn
1805         {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1806     }
1807     \stex_smsmode_do:
1808     \ignorespacesandpars
1809 }

```

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

1810 `\endpackage`

Chapter 30

STEX -Symbols Implementation

```
1811 <*package>
1812
1813 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1814
      Warnings and error messages
1815 \msg_new:nnn{stex}{error/wrongargs}{
1816   args~value~in~symbol~declaration~for~#1~
1817   needs~to~be~i,~a,~b~or~B,~but~#2~given
1818 }
1819 \msg_new:nnn{stex}{error/unknownsymbol}{
1820   No~symbol~#1~found!
1821 }
1822 \msg_new:nnn{stex}{error/seqlength}{
1823   Expected~#1~arguments;~got~#2!
1824 }
```

30.1 Symbol Declarations

```
1825 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols

1826 \cs_new_protected:Nn \stex_all_symbols:n {
1827   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1828   \seq_map_inline:Nn \l_stex_all_modules_seq {
1829     \seq_map_inline:cn{c_stex_module_##1_constants}{
1830       \__stex_symdecl_all_symbols_cs{##1?####1}
1831     }
1832   }
1833 }

(End definition for \stex_all_symbols:n. This function is documented on page 37.)

\STEXsymbol

1834 \NewDocumentCommand \STEXsymbol { m } {
1835   \stex_get_symbol:n { #1 }
```

```

1836 \exp_args:No
1837 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1838 }

```

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

`symdecl` arguments:

```

1839 \keys_define:nn { stex / symdecl } {
1840   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1841   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1842   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1843   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1844   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1845   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1846   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1847   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1848   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1849   assoc     .choices:nn =
1850     {bin,binl,binr,pre,conj,pwconj}
1851     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1852 }
1853
1854 \bool_new:N \l_stex_symdecl_make_macro_bool
1855
1856 \cs_new_protected:Nn \__stex_symdecl_args:n {
1857   \str_clear:N \l_stex_symdecl_name_str
1858   \str_clear:N \l_stex_symdecl_args_str
1859   \str_clear:N \l_stex_symdecl_deprecate_str
1860   \str_clear:N \l_stex_symdecl_astype_str
1861   \bool_set_false:N \l_stex_symdecl_local_bool
1862   \tl_clear:N \l_stex_symdecl_type_tl
1863   \tl_clear:N \l_stex_symdecl_definiens_tl
1864
1865   \keys_set:nn { stex / symdecl } { #1 }
1866 }

```

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

```

1867
1868 \NewDocumentCommand \symdecl { s m O{} } {
1869   \__stex_symdecl_args:n { #3 }
1870   \IfBooleanTF #1 {
1871     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1872   } {
1873     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1874   }
1875   \stex_symdecl_do:n { #2 }
1876   \stex_smsmode_do:
1877 }
1878
1879 \cs_new_protected:Nn \stex_symdecl_do:nn {
1880   \__stex_symdecl_args:n{#1}
1881   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1882   \stex_symdecl_do:n{#2}
1883 }

```

```

1884
1885 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1886 \cs_new_protected:Nn \stex_symdecl_do:n {
1887   \stex_if_in_module:F {
1888     % TODO throw error? some default namespace?
1889   }
1890
1891   \str_if_empty:NT \l_stex_symdecl_name_str {
1892     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1893   }
1894
1895   \prop_if_exist:cT { l_stex_symdecl_
1896     \l_stex_current_module_str ?
1897     \l_stex_symdecl_name_str
1898   }_prop
1899   {
1900     % TODO throw error (beware of circular dependencies)
1901   }
1902
1903   \prop_clear:N \l_tmpa_prop
1904   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1905   \seq_clear:N \l_tmpa_seq
1906   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1907   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1908
1909   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1910     \str_if_empty:NF \l_stex_module_deprecate_str {
1911       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1912     }
1913   }
1914   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1915
1916   \exp_args:No \stex_add_constant_to_current_module:n {
1917     \l_stex_symdecl_name_str
1918   }
1919
1920   % arity/args
1921   \int_zero:N \l_tmpb_int
1922
1923   \bool_set_true:N \l_tmpa_bool
1924   \str_map_inline:Nn \l_stex_symdecl_args_str {
1925     \token_case_meaning:NnF ##1 {
1926       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1927       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1928       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1929       {\tl_to_str:n a} {
1930         \bool_set_false:N \l_tmpa_bool
1931         \int_incr:N \l_tmpb_int
1932       }
1933       {\tl_to_str:n B} {

```

```

1934         \bool_set_false:N \l_tmpa_bool
1935         \int_incr:N \l_tmpb_int
1936     }
1937 }{
1938     \msg_error:nnxx{stex}{error/wrongargs}{
1939         \l_stex_current_module_str ?
1940         \l_stex_symdecl_name_str
1941     }{##1}
1942 }
1943 }
1944 \bool_if:NTF \l_tmpa_bool {
1945     % possibly numeric
1946     \str_if_empty:NTF \l_stex_symdecl_args_str {
1947         \prop_put:Nnn \l_tmpa_prop { args } {}
1948         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1949     }{
1950         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1951         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1952         \str_clear:N \l_tmpa_str
1953         \int_step_inline:nn \l_tmpa_int {
1954             \str_put_right:Nn \l_tmpa_str i
1955         }
1956         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1957     }
1958 } {
1959     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1960     \prop_put:Nnx \l_tmpa_prop { arity }
1961         { \str_count:N \l_stex_symdecl_args_str }
1962 }
1963 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1964
1965 % semantic macro
1966
1967 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1968     \exp_args:Nx \stex_do_up_to_module:n {
1969         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1970             \l_stex_current_module_str ? \l_stex_symdecl_name_str
1971         }}
1972     }
1973 }
1974
1975 \bool_if:NF \l_stex_symdecl_local_bool {
1976     \exp_args:Nx \stex_add_to_current_module:n {
1977         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1978             \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979         } }
1980     }
1981 }
1982 }
1983
1984 \stex_debug:nn{symbols}{New~symbol:~
1985     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1986     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1987     Args:~\prop_item:Nn \l_tmpa_prop { args }^^J

```

```

1988     Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
1989 }
1990
1991 % circular dependencies require this:
1992
1993 \prop_if_exist:cF {
1994     \l_stex_symdecl_
1995     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1996     _prop
1997 } {
1998     \exp_args:Nx \stex_do_up_to_module:n {
1999         \prop_set_from_keyval:cn {
2000             \l_stex_symdecl_
2001             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002             _prop
2003         } {\prop_to_keyval:N \l_tmpa_prop}
2004     }
2005 }
2006
2007 \seq_clear:c {
2008     \l_stex_symdecl_
2009     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2010     _notations
2011 }
2012
2013 \bool_if:NF \l_stex_symdecl_local_bool {
2014     \exp_args:Nx
2015     \stex_add_to_current_module:n {
2016         \seq_clear:c {
2017             \l_stex_symdecl_
2018             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2019             _notations
2020         }
2021         \prop_set_from_keyval:cn {
2022             \l_stex_symdecl_
2023             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2024             _prop
2025         } {
2026             name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2027             module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2028             type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2029             args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2030             arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2031             assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2032         }
2033     }
2034 }
2035
2036 \stex_if_smsmode:F {
2037 %     \exp_args:Nx \stex_do_up_to_module:n {
2038 %         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2039 %             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2040 %         }
2041 %     }

```

```

2042 \stex_if_do_html:T {
2043   \stex_annotate_invisible:nnn {symdecl} {
2044     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2045   } {
2046     \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2047     \stex_annotate_invisible:nnn{args}{}}{
2048       \prop_item:Nn \l_tmpa_prop { args }
2049     }
2050     \stex_annotate_invisible:nnn{macroname}{#1}{}
2051     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2052       \stex_annotate_invisible:nnn{definiens}{}
2053       {\l_stex_symdecl_definiens_tl$}
2054     }
2055     \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2056       \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2057     }
2058   }
2059 }
2060 }
2061 }

```

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

`\stex_get_symbol:n`

```

2062 \str_new:N \l_stex_get_symbol_uri_str
2063
2064 \cs_new_protected:Nn \stex_get_symbol:n {
2065   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2066     \tl_set:Nn \l_tmpa_tl { #1 }
2067     \__stex_symdecl_get_symbol_from_cs:
2068   }{
2069     % argument is a string
2070     % is it a command name?
2071     \cs_if_exist:cTF { #1 }{
2072       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2073       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2074       \str_if_empty:NTF \l_tmpa_str {
2075         \exp_args:Nx \cs_if_eq:NNTF {
2076           \tl_head:N \l_tmpa_tl
2077         } \stex_invoke_symbol:n {
2078           \__stex_symdecl_get_symbol_from_cs:
2079         }{
2080           \__stex_symdecl_get_symbol_from_string:n { #1 }
2081         }
2082       } {
2083         \__stex_symdecl_get_symbol_from_string:n { #1 }
2084       }
2085     }{
2086       % argument is not a command name
2087       \__stex_symdecl_get_symbol_from_string:n { #1 }
2088       % \l_stex_all_symbols_seq
2089     }
2090   }
2091   \str_if_eq:eeF {

```



```

2092     \prop_item:cn {
2093       l_stex_symdecl~\l_stex_get_symbol_uri_str _prop
2094     }{ deprecate }
2095   }{}{
2096     \msg_warning:nnxx{stex}{warning/deprecated}{
2097       Symbol~\l_stex_get_symbol_uri_str
2098     }{
2099       \prop_item:cn {l_stex_symdecl~\l_stex_get_symbol_uri_str _prop}{ deprecate }
2100     }
2101   }
2102 }
2103
2104 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2105   \tl_set:Nn \l_tmpa_tl {
2106     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2107   }
2108   \str_set:Nn \l_tmpa_str { #1 }
2109   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2110
2111   \stex_all_symbols:n {
2112     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2113       \seq_map_break:n{\seq_map_break:n{
2114         \tl_set:Nn \l_tmpa_tl {
2115           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2116         }
2117       }}
2118     }
2119   }
2120
2121   \l_tmpa_tl
2122 }
2123
2124 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2125   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2126   { \tl_tail:N \l_tmpa_tl }
2127   \tl_if_single:NTF \l_tmpa_tl {
2128     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2129       \exp_after:wN \str_set:Nn \exp_after:wN
2130         \l_stex_get_symbol_uri_str \l_tmpa_tl
2131     }{
2132       % TODO
2133       % tail is not a single group
2134     }
2135   }{
2136     % TODO
2137     % tail is not a single group
2138   }
2139 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page [37](#).)

30.2 Notations

2140 `<@@=stex_notation>`

```

notation arguments:
2141 \keys_define:nn { stex / notation } {
2142   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2143   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2144   prec .str_set_x:N = \l__stex_notation_prec_str ,
2145   op .tl_set:N = \l__stex_notation_op_tl ,
2146   primary .bool_set:N = \l__stex_notation_primary_bool ,
2147   primary .default:n = {true} ,
2148   unknown .code:n = \str_set:Nx
2149     \l__stex_notation_variant_str \l_keys_key_str
2150 }
2151
2152 \cs_new_protected:Nn \_stex_notation_args:n {
2153   \str_clear:N \l__stex_notation_lang_str
2154   \str_clear:N \l__stex_notation_variant_str
2155   \str_clear:N \l__stex_notation_prec_str
2156   \tl_clear:N \l__stex_notation_op_tl
2157   \bool_set_false:N \l__stex_notation_primary_bool
2158
2159   \keys_set:nn { stex / notation } { #1 }
2160 }

```

\notation

```

2161 \NewDocumentCommand \notation { s m O{}} {
2162   \_stex_notation_args:n { #3 }
2163   \tl_clear:N \l_stex_symdecl_definiens_tl
2164   \stex_get_symbol:n { #2 }
2165   \tl_set:Nn \l_stex_notation_after_do_tl {
2166     \__stex_notation_final:
2167     \IfBooleanTF#1{
2168       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2169     }{}
2170     \stex_smsmode_do:\ignorespacesandpars
2171   }
2172   \stex_notation_do:nnnnn
2173     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2174     { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2175     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2176     { \l__stex_notation_prec_str }
2177   }
2178   \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nnnnn

```

2179 \seq_new:N \l__stex_notation_precedences_seq
2180 \tl_new:N \l__stex_notation_opprec_tl
2181 \int_new:N \l__stex_notation_currarg_int
2182 \tl_new:N \stex_symbol_after_invokation_tl
2183
2184 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2185   \let\l_stex_current_symbol_str\relax
2186   \seq_clear:N \l__stex_notation_precedences_seq
2187   \tl_clear:N \l__stex_notation_opprec_tl
2188   \str_set:Nx \l__stex_notation_args_str { #1 }

```

```

2189 \str_set:Nx \l__stex_notation_arity_str { #2 }
2190 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2191 \str_set:Nx \l__stex_notation_prec_str { #4 }
2192
2193 % precedences
2194 \str_if_empty:NTF \l__stex_notation_prec_str {
2195   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2196     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2197   }{
2198     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2199   }
2200 } {
2201   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2202     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2203     \int_step_inline:nn { \l__stex_notation_arity_str } {
2204       \exp_args:NNo
2205       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2206     }
2207   }{
2208     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2209     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2210       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2211       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2212         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2213         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2214         \seq_map_inline:Nn \l_tmpa_seq {
2215           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2216         }
2217       }
2218     }{
2219       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2220         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2221       }{
2222         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2223       }
2224     }
2225   }
2226 }
2227
2228 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2229 \int_step_inline:nn { \l__stex_notation_arity_str } {
2230   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2231     \exp_args:NNo
2232     \seq_put_right:No \l__stex_notation_precedences_seq {
2233       \l__stex_notation_opprec_tl
2234     }
2235   }
2236 }
2237 \tl_clear:N \l_stex_notation_dummyargs_tl
2238
2239 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2240   \exp_args:NNe
2241   \cs_set:Npn \l_stex_notation_macrocode_cs {
2242     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }

```

```

2243     { \l__stex_notation_suffix_str }
2244     { \l__stex_notation_opprec_tl }
2245     { \exp_not:n { #5 } }
2246   }
2247   \l_stex_notation_after_do_tl
2248 }{
2249   \str_if_in:NnTF \l__stex_notation_args_str b {
2250     \exp_args:Nne \use:nn
2251     {
2252       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2253       \cs_set:Npn \l__stex_notation_arity_str } { {
2254         \_stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2255         { \l__stex_notation_suffix_str }
2256         { \l__stex_notation_opprec_tl }
2257         { \exp_not:n { #5 } }
2258       } }
2259     }{
2260       \str_if_in:NnTF \l__stex_notation_args_str B {
2261         \exp_args:Nne \use:nn
2262         {
2263           \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2264           \cs_set:Npn \l__stex_notation_arity_str } { {
2265             \_stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2266             { \l__stex_notation_suffix_str }
2267             { \l__stex_notation_opprec_tl }
2268             { \exp_not:n { #5 } }
2269           } }
2270         }{
2271           \exp_args:Nne \use:nn
2272           {
2273             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2274             \cs_set:Npn \l__stex_notation_arity_str } { {
2275               \_stex_term_math_oma:nmmm { \l_stex_current_symbol_str }
2276               { \l__stex_notation_suffix_str }
2277               { \l__stex_notation_opprec_tl }
2278               { \exp_not:n { #5 } }
2279             } }
2280           }
2281         }
2282       }
2283       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2284       \int_zero:N \l__stex_notation_currarg_int
2285       \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2286       \__stex_notation_arguments:
2287     }
2288   }

```

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

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

```

2289 \cs_new_protected:Nn \__stex_notation_arguments: {
2290   \int_incr:N \l__stex_notation_currarg_int
2291   \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2292     \l_stex_notation_after_do_tl

```

```

2293 }{
2294   \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2295   \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2296   \str_if_eq:VnTF \l_tmpa_str a {
2297     \__stex_notation_argument_assoc:n
2298   }{
2299     \str_if_eq:VnTF \l_tmpa_str B {
2300       \__stex_notation_argument_assoc:n
2301     }{
2302       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2303       \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2304         { \stex_term_math_arg:nnn
2305           { \int_use:N \l__stex_notation_currarg_int }
2306           { \l_tmpa_str }
2307           { ###\int_use:N \l__stex_notation_currarg_int }
2308         }
2309       }
2310       \__stex_notation_arguments:
2311     }
2312   }
2313 }
2314 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2315 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2316
2317   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2318     {\l__stex_notation_arity_str}{
2319     #1
2320   }
2321   \int_zero:N \l_tmpa_int
2322   \tl_clear:N \l_tmpa_tl
2323   \str_map_inline:Nn \l__stex_notation_args_str {
2324     \int_incr:N \l_tmpa_int
2325     \tl_put_right:Nx \l_tmpa_tl {
2326       \str_if_eq:nnTF {##1}{a}{ {} }{
2327         \str_if_eq:nnTF {##1}{B}{ {} }{
2328           {\stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2329         }
2330       }
2331     }
2332   }
2333   \exp_after:wN\exp_after:wN\exp_after:wN \def
2334   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2335   \exp_after:wN\exp_after:wN\exp_after:wN ##
2336   \exp_after:wN\exp_after:wN\exp_after:wN 1
2337   \exp_after:wN\exp_after:wN\exp_after:wN ##
2338   \exp_after:wN\exp_after:wN\exp_after:wN 2
2339   \exp_after:wN\exp_after:wN\exp_after:wN {
2340     \exp_after:wN \exp_after:wN \exp_after:wN
2341     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2342     \exp_after:wN \l_tmpa_cs \l_tmpa_tl

```

```

2343     }
2344   }
2345
2346   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2347   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2348     \stex_term_math_assoc_arg:nnnn
2349     { \int_use:N \l__stex_notation_currarg_int }
2350     { \l_tmpa_str }
2351     { ####\int_use:N \l__stex_notation_currarg_int }
2352     { \l_tmpa_cs {####1} {####2} }
2353   } }
2354   \__stex_notation_arguments:
2355 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2356 \cs_new_protected:Nn \__stex_notation_final: {
2357   \exp_args:Nne \use:nn
2358   {
2359     \cs_generate_from_arg_count:cNnn {
2360       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2361       \l__stex_notation_suffix_str
2362       _cs
2363     }
2364     \cs_set:Npn \l__stex_notation_arity_str { { {
2365       \exp_after:wN \exp_after:wN \exp_after:wN
2366       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2367       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2368     } } }
2369
2370     \tl_if_empty:NF \l__stex_notation_op_tl {
2371       \cs_set:cpx {
2372         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2373         \l__stex_notation_suffix_str
2374         _cs
2375       } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2376     }
2377
2378     \exp_args:Ne
2379     \stex_add_to_current_module:n {
2380       \cs_generate_from_arg_count:cNnn {
2381         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2382         \l__stex_notation_suffix_str
2383         _cs
2384       } \cs_set:Npn { \l__stex_notation_arity_str } {
2385         \exp_after:wN \exp_after:wN \exp_after:wN
2386         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2387         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2388       }
2389       \tl_if_empty:NF \l__stex_notation_op_tl {
2390         \cs_set:cpn {
2391           stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2392           \l__stex_notation_suffix_str

```

```

2393     _cs
2394   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2395 }
2396 }
2397 %\exp_args:Nx
2398 % \stex_do_up_to_module:n {
2399   \seq_put_right:cx {
2400     \l_stex_symdecl_ \l_stex_get_symbol_uri_str
2401     _notations
2402   } {
2403     \l__stex_notation_suffix_str
2404   }
2405 % }
2406
2407 \stex_debug:nn{symbols}{
2408   Notation~\l__stex_notation_suffix_str
2409   ~for~\l_stex_get_symbol_uri_str^^J
2410   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2411   Argument~precedences:~
2412   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2413   Notation: \cs_meaning:c {
2414     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2415     \l__stex_notation_suffix_str
2416     _cs
2417   }
2418 }
2419
2420 \exp_args:Ne
2421 \stex_add_to_current_module:n {
2422   \seq_put_right:cn {
2423     \l_stex_symdecl_ \l_stex_get_symbol_uri_str
2424     _notations
2425   } { \l__stex_notation_suffix_str }
2426 }
2427
2428 \stex_if_smsmode:F {
2429
2430   % HTML annotations
2431   \stex_if_do_html:T {
2432     \stex_annotate_invisible:nnn { notation }
2433     { \l_stex_get_symbol_uri_str } {
2434       \stex_annotate_invisible:nnn { notationfragment }
2435       { \l__stex_notation_suffix_str }{}
2436     }
2437     \stex_annotate_invisible:nnn { precedence }
2438     { \l__stex_notation_prec_str }{}
2439
2440     \int_zero:N \l_tmpa_int
2441     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2442     \tl_clear:N \l_tmpa_tl
2443     \int_step_inline:nn { \l__stex_notation_arity_str }{
2444       \int_incr:N \l_tmpa_int
2445       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2446       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r

```

```

2447         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2448             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2449             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2450         } }
2451     }{
2452         \str_if_eq:VnTF \l_tmpb_str B {
2453             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2454                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2455                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2456             } }
2457         }{
2458             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2459                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2460             } }
2461         }
2462     }
2463 }
2464 \stex_annotate_invisible:nnn { notationcomp }{}{
2465     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2466     $ \exp_args:Nno \use:nn { \use:c {
2467         stex_notation_ \l_stex_current_symbol_str
2468         \c_hash_str \l__stex_notation_suffix_str _cs
2469     } } { \l_tmpa_tl } $
2470 }
2471 }
2472 }
2473 }
2474 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

2475 \keys_define:nn { stex / setnotation } {
2476     lang .tl_set_x:N = \l__stex_notation_lang_str ,
2477     variant .tl_set_x:N = \l__stex_notation_variant_str ,
2478     unknown .code:n = \str_set:Nx
2479         \l__stex_notation_variant_str \l_keys_key_str
2480 }
2481
2482 \cs_new_protected:Nn \stex_setnotation_args:n {
2483     \str_clear:N \l__stex_notation_lang_str
2484     \str_clear:N \l__stex_notation_variant_str
2485     \keys_set:nn { stex / setnotation } { #1 }
2486 }
2487
2488 \cs_new_protected:Nn \stex_setnotation:n {
2489     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2490     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2491         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2492         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2493         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2494         { \c_hash_str }
2495         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2496         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```



```

2497 \exp_args:Nx \stex_add_to_current_module:n {
2498   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2499   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2500   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2501   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2502   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2503   { \c_hash_str }
2504 }
2505 \stex_debug:nn {notations}{
2506   Setting~default~notation~
2507   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2508   #1 \\
2509   \expandafter\meaning\csname
2510   l_stex_symdecl_#1 _notations\endcsname
2511 }
2512 }{
2513   % todo throw error
2514 }
2515 }
2516
2517 \NewDocumentCommand \setnotation {m m} {
2518   \stex_get_symbol:n { #1 }
2519   \_stex_setnotation_args:n { #2 }
2520   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2521   \stex_smsmode_do:\ignorespacesandpars
2522 }
2523
2524 \cs_new_protected:Nn \stex_copy_notations:nn {
2525   \stex_debug:nn {notations}{
2526     Copying~notations~from~#2~to~#1\\
2527     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2528   }
2529   \tl_clear:N \l_tmpa_tl
2530   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2531     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2532 }
2533 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2534   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2535   \edef \l_tmpa_tl {
2536     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2537     \exp_after:wN\exp_after:wN\exp_after:wN {
2538       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2539     }
2540   }
2541   \exp_args:Nx
2542   \stex_do_up_to_module:n {
2543     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2544     \cs_generate_from_arg_count:cNnn {
2545       stex_notation_ #1 \c_hash_str ##1 _cs
2546     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2547       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2548     }
2549   }
2550 }

```

```

2551 }
2552
2553 \NewDocumentCommand \copynotation {m m} {
2554   \stex_get_symbol:n { #1 }
2555   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2556   \stex_get_symbol:n { #2 }
2557   \exp_args:Noo
2558   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2559   \exp_args:Nx \stex_add_import_to_current_module:n{
2560     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2561   }
2562   \stex_smsmode_do:\ignorespacesandpars
2563 }
2564

```

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

\symdef

```

2565 \keys_define:nn { stex / symdef } {
2566   name .str_set_x:N = \l_stex_symdecl_name_str ,
2567   local .bool_set:N = \l_stex_symdecl_local_bool ,
2568   args .str_set_x:N = \l_stex_symdecl_args_str ,
2569   type .tl_set:N = \l_stex_symdecl_type_tl ,
2570   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2571   op .tl_set:N = \l__stex_notation_op_tl ,
2572   lang .str_set_x:N = \l__stex_notation_lang_str ,
2573   variant .str_set_x:N = \l__stex_notation_variant_str ,
2574   prec .str_set_x:N = \l__stex_notation_prec_str ,
2575   assoc .choices:nn =
2576     {bin,binl,binr,pre,conj,pwconj}
2577     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2578   unknown .code:n = \str_set:Nx
2579     \l__stex_notation_variant_str \l_keys_key_str
2580 }
2581
2582 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2583   \str_clear:N \l_stex_symdecl_name_str
2584   \str_clear:N \l_stex_symdecl_args_str
2585   \str_clear:N \l_stex_symdecl_assoctype_str
2586   \bool_set_false:N \l_stex_symdecl_local_bool
2587   \tl_clear:N \l_stex_symdecl_type_tl
2588   \tl_clear:N \l_stex_symdecl_definiens_tl
2589   \str_clear:N \l__stex_notation_lang_str
2590   \str_clear:N \l__stex_notation_variant_str
2591   \str_clear:N \l__stex_notation_prec_str
2592   \tl_clear:N \l__stex_notation_op_tl
2593
2594   \keys_set:nn { stex / symdef } { #1 }
2595 }
2596
2597 \NewDocumentCommand \symdef { m O{} } {
2598   \__stex_notation_symdef_args:n { #2 }
2599   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2600   \stex_symdecl_do:n { #1 }

```

```

2601 \tl_set:Nn \l_stex_notation_after_do_tl {
2602   \__stex_notation_final:
2603   \stex_smsmode_do:\ignorespacesandpars
2604 }
2605 \str_set:Nx \l_stex_get_symbol_uri_str {
2606   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2607 }
2608 \exp_args:Nx \stex_notation_do:nnnnn
2609   { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} {args} }
2610   { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} {arity} }
2611   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2612   { \l__stex_notation_prec_str}
2613 }
2614 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

30.3 Variables

```

2615 <@@=stex_variables>
2616
2617 \keys_define:nn { stex / vardef } {
2618   name      .str_set:N = \l__stex_variables_name_str ,
2619   args      .str_set:N = \l__stex_variables_args_str ,
2620   type      .tl_set:N  = \l__stex_variables_type_tl ,
2621   def       .tl_set:N  = \l__stex_variables_def_tl ,
2622   op        .tl_set:N  = \l__stex_variables_op_tl ,
2623   prec      .str_set:N = \l__stex_variables_prec_str ,
2624   assoc     .choices:nn =
2625     {bin,binl,binr,pre,conj,pwconj}
2626     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2627   bind      .choices:nn =
2628     {forall,exists}
2629     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2630 }
2631
2632 \cs_new_protected:Nn \__stex_variables_args:n {
2633   \str_clear:N \l__stex_variables_name_str
2634   \str_clear:N \l__stex_variables_args_str
2635   \str_clear:N \l__stex_variables_prec_str
2636   \str_clear:N \l__stex_variables_assoctype_str
2637   \str_clear:N \l__stex_variables_bind_str
2638   \tl_clear:N \l__stex_variables_type_tl
2639   \tl_clear:N \l__stex_variables_def_tl
2640   \tl_clear:N \l__stex_variables_op_tl
2641
2642   \keys_set:nn { stex / vardef } { #1 }
2643 }
2644
2645 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2646   \__stex_variables_args:n {#2}
2647   \str_if_empty:NT \l__stex_variables_name_str {
2648     \str_set:Nx \l__stex_variables_name_str { #1 }
2649   }

```

```

2650 \prop_clear:N \l_tmpa_prop
2651 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2652
2653 \int_zero:N \l_tmpb_int
2654 \bool_set_true:N \l_tmpa_bool
2655 \str_map_inline:Nn \l__stex_variables_args_str {
2656   \token_case_meaning:NnF ##1 {
2657     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2658     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2659     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2660     {\tl_to_str:n a} {
2661       \bool_set_false:N \l_tmpa_bool
2662       \int_incr:N \l_tmpb_int
2663     }
2664     {\tl_to_str:n B} {
2665       \bool_set_false:N \l_tmpa_bool
2666       \int_incr:N \l_tmpb_int
2667     }
2668   }{
2669     \msg_error:nxxx{stex}{error/wrongargs}{
2670       variable~\l__stex_variables_name_str
2671     }{##1}
2672   }
2673 }
2674 \bool_if:NTF \l_tmpa_bool {
2675   % possibly numeric
2676   \str_if_empty:NTF \l__stex_variables_args_str {
2677     \prop_put:Nnn \l_tmpa_prop { args } {}
2678     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2679   }{
2680     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2681     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2682     \str_clear:N \l_tmpa_str
2683     \int_step_inline:nn \l_tmpa_int {
2684       \str_put_right:Nn \l_tmpa_str i
2685     }
2686     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2687     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2688   }
2689 } {
2690   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2691   \prop_put:Nnx \l_tmpa_prop { arity }
2692   { \str_count:N \l__stex_variables_args_str }
2693 }
2694 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2695 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2696
2697 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2698
2699 \tl_if_empty:NF \l__stex_variables_op_tl {
2700   \cs_set:cpx {
2701     stex_var_op_notation_ \l__stex_variables_name_str _cs
2702   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2703 }

```

```

2704 \tl_set:Nn \l_stex_notation_after_do_tl {
2705   \exp_args:Nne \use:nn {
2706     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2707     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2708   } {{
2709     \exp_after:wN \exp_after:wN \exp_after:wN
2710     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2711     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2712     }}
2713   \stex_if_do_html:T {
2714     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2715       \stex_annotate_invisible:nnn { precedence }
2716       { \l__stex_variables_prec_str }{}
2717       \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l__stex_variables_type_str }
2718       \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2719       \stex_annotate_invisible:nnn{macroname}{#1}{}
2720       \tl_if_empty:NF \l__stex_variables_def_tl {
2721         \stex_annotate_invisible:nnn{definiens}{}
2722         { $\l__stex_variables_def_tl$ }
2723       }
2724     }
2725     \str_if_empty:NF \l__stex_variables_assoctype_str {
2726       \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2727     }
2728     \int_zero:N \l_tmpa_int
2729     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2730     \tl_clear:N \l_tmpa_tl
2731     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{{
2732       \int_incr:N \l_tmpa_int
2733       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2734       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2735       \str_if_eq:VnTF \l_tmpb_str a {
2736         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2737           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2738           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2739         } }
2740       }{
2741         \str_if_eq:VnTF \l_tmpb_str B {
2742           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2743             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2744             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2745           } }
2746         }{
2747           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2748             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2749           } }
2750         }
2751       }
2752     }}
2753     \stex_annotate_invisible:nnn { notationcomp }{}{
2754       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2755       $ \exp_args:Nno \use:nn { \use:c {
2756         stex_var_notation_\l__stex_variables_name_str_cs
2757       } } { \l_tmpa_tl } $

```

```

2758     }
2759   }
2760   }\ignorespacesandpars
2761 }
2762
2763 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2764 }
2765
2766 \cs_new:Nn \_stex_reset:N {
2767   \tl_if_exist:NTF #1 {
2768     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2769   }{
2770     \let \exp_not:N #1 \exp_not:N \undefined
2771   }
2772 }
2773
2774 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2775   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2776   \exp_args:Nnx \use:nn {
2777     % TODO
2778     \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2779       #2
2780     }
2781   }{
2782     \_stex_reset:N \varnot
2783     \_stex_reset:N \vartype
2784     \_stex_reset:N \vardefi
2785   }
2786 }
2787
2788 \NewDocumentCommand \vardef { s } {
2789   \IfBooleanTF#1 {
2790     \__stex_variables_do_complex:nn
2791   }{
2792     \__stex_variables_do_simple:nnn
2793   }
2794 }
2795
2796 \NewDocumentCommand \svar { 0{} m }{
2797   \tl_if_empty:nTF {#1}{
2798     \str_set:Nn \l_tmpa_str { #2 }
2799   }{
2800     \str_set:Nn \l_tmpa_str { #1 }
2801   }
2802   \_stex_term_omv:nn {
2803     var://\l_tmpa_str
2804   }{ \comp{ #2 } }
2805 }
2806
2807
2808
2809 \keys_define:nn { stex / varseq } {
2810   name      .str_set_x:N = \l__stex_variables_name_str ,
2811   args      .int_set:N   = \l__stex_variables_args_int ,

```

```

2812 type .tl_set:N = \l__stex_variables_type_tl ,
2813 mid .tl_set:N = \l__stex_variables_mid_tl ,
2814 bind .choices:nn =
2815 {forall,exists}
2816 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2817 }
2818
2819 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2820 \str_clear:N \l__stex_variables_name_str
2821 \int_set:Nn \l__stex_variables_args_int 1
2822 \tl_clear:N \l__stex_variables_type_tl
2823 \str_clear:N \l__stex_variables_bind_str
2824
2825 \keys_set:nn { stex / varseq } { #1 }
2826 }
2827
2828 \NewDocumentCommand \varseq {m O{} m m m}{
2829 \__stex_variables_seq_args:n { #2 }
2830 \str_if_empty:NT \l__stex_variables_name_str {
2831 \str_set:Nx \l__stex_variables_name_str { #1 }
2832 }
2833 \prop_clear:N \l_tmpa_prop
2834 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2835
2836 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2837 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2838 \msg_error:nnxx{stex}{error/seqlength}
2839 {\int_use:N \l__stex_variables_args_int}
2840 {\seq_count:N \l_tmpa_seq}
2841 }
2842 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2843 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2844 \msg_error:nnxx{stex}{error/seqlength}
2845 {\int_use:N \l__stex_variables_args_int}
2846 {\seq_count:N \l_tmpb_seq}
2847 }
2848 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2849 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2850
2851 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
2852 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
2853
2854 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2855 \int_step_inline:nn \l__stex_variables_args_int {
2856 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2857 }
2858 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2859 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2860 \tl_if_empty:NF \l__stex_variables_mid_tl {
2861 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2862 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2863 }
2864 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2865 \int_step_inline:nn \l__stex_variables_args_int {

```

```

2866 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2867 }
2868 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2869 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2870
2871
2872 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2873
2874 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2875
2876 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2877
2878 \int_step_inline:nn \l__stex_variables_args_int {
2879   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2880     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2881   }}
2882 }
2883
2884 \tl_set:Nx \l_tmpa_tl {
2885   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
2886     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2887   }
2888 }
2889
2890 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2891
2892 \exp_args:Nno \use:nn {
2893   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2894   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
2895
2896   \stex_debug:nn{sequences}{New~Sequence:~
2897     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2898     \prop_to_keyval:N \l_tmpa_prop
2899   }
2900
2901   \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2902   \ignorespacesandpars
2903 }
2904
2905 \end{package}

```


Chapter 31

STEX -Terms Implementation

```
2906 <*package>
2907
2908 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2909
2910 <@@=stex_terms>
2911
2912 Warnings and error messages
2913 \msg_new:nnn{stex}{error/nonotation}{
2914   Symbol~#1~invoked,~but~has~no~notation#2!
2915 }
2916 \msg_new:nnn{stex}{error/notationarg}{
2917   Error~in~parsing~notation~#1
2918 }
2919 \msg_new:nnn{stex}{error/noop}{
2920   Symbol~#1~has~no~operator~notation~for~notation~#2
2921 }
2922 \msg_new:nnn{stex}{error/notallowed}{
2923   Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
2924 }
2925
```

31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2924
2925
2926 \bool_new:N \l_stex_allow_semantic_bool
2927 \bool_set_true:N \l_stex_allow_semantic_bool
2928
2929 \cs_new_protected:Nn \stex_invoke_symbol:n {
2930   \bool_if:NTF \l_stex_allow_semantic_bool {
2931     \str_if_eq:eeF {
2932       \prop_item:cn {
2933         l_stex_symdecl_#1_prop
2934       }{ deprecate }
2935     }
2936   }
2937 }
```

```

2935   }{}{
2936     \msg_warning:nxxx{stex}{warning/deprecated}{
2937       Symbol~#1
2938     }{
2939       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2940     }
2941   }
2942   \if_mode_math:
2943     \exp_after:wN \__stex_terms_invoke_math:n
2944   \else:
2945     \exp_after:wN \__stex_terms_invoke_text:n
2946   \fi: { #1 }
2947 }{
2948   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2949 }
2950 }
2951
2952 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2953   \peek_charcode_remove:NTF ! {
2954     \__stex_terms_invoke_op_custom:nn {#1}
2955   }{
2956     \__stex_terms_invoke_custom:nn {#1}
2957   }
2958 }
2959
2960 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2961   \peek_charcode_remove:NTF ! {
2962     % operator
2963     \peek_charcode_remove:NTF * {
2964       % custom op
2965       \__stex_terms_invoke_op_custom:nn {#1}
2966     }{
2967       % op notation
2968       \peek_charcode:NTF [ {
2969         \__stex_terms_invoke_op_notation:nw {#1}
2970       }{
2971         \__stex_terms_invoke_op_notation:nw {#1}[]
2972       }
2973     }
2974   }{
2975     \peek_charcode_remove:NTF * {
2976       \__stex_terms_invoke_custom:nn {#1}
2977       % custom
2978     }{
2979       % normal
2980       \peek_charcode:NTF [ {
2981         \__stex_terms_invoke_notation:nw {#1}
2982       }{
2983         \__stex_terms_invoke_notation:nw {#1}[]
2984       }
2985     }
2986   }
2987 }
2988

```

```

2989
2990 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2991   \exp_args:Nnx \use:nn {
2992     \def\comp{\_comp}
2993     \str_set:Nn \l_stex_current_symbol_str { #1 }
2994     \bool_set_false:N \l_stex_allow_semantic_bool
2995     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2996       \comp{ #2 }
2997     }
2998   }{
2999     \stex_reset:N \comp
3000     \stex_reset:N \l_stex_current_symbol_str
3001     \bool_set_true:N \l_stex_allow_semantic_bool
3002   }
3003 }
3004
3005 \keys_define:nn { stex / terms } {
3006   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3007   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3008   unknown .code:n = \str_set:Nx
3009     \l_stex_notation_variant_str \l_keys_key_str
3010 }
3011
3012 \cs_new_protected:Nn \__stex_terms_args:n {
3013   \str_clear:N \l_stex_notation_lang_str
3014   \str_clear:N \l_stex_notation_variant_str
3015
3016   \keys_set:nn { stex / terms } { #1 }
3017 }
3018
3019 \cs_new_protected:Nn \stex_find_notation:nn {
3020   \__stex_terms_args:n { #2 }
3021   \seq_if_empty:cTF {
3022     l_stex_symdecl_ #1 _notations
3023   } {
3024     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3025   } {
3026     \bool_lazy_all:nTF {
3027       {\str_if_empty_p:N \l_stex_notation_variant_str}
3028       {\str_if_empty_p:N \l_stex_notation_lang_str}
3029     }{
3030       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3031     }{
3032       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3033         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3034       }{
3035         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3036       }{
3037         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3038           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3039         }
3040       }
3041     }
3042   }

```

```

3043 }
3044
3045 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3046   \exp_args:Nnx \use:nn {
3047     \def\comp{\_comp}
3048     \str_set:Nn \l_stex_current_symbol_str { #1 }
3049     \stex_find_notation:nn { #1 }{ #2 }
3050     \bool_set_false:N \l_stex_allow_semantic_bool
3051     \cs_if_exist:cTF {
3052       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3053     }{
3054       \_stex_term_oms:nnn {
3055         #1 \c_hash_str \l_stex_notation_variant_str
3056       }{ #1 }{
3057         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3058       }
3059     }{
3060       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3061         \cs_if_exist:cTF {
3062           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3063         }{
3064           \tl_set:Nx \stex_symbol_after_invokation_tl {
3065             \_stex_reset:N \comp
3066             \_stex_reset:N \stex_symbol_after_invokation_tl
3067             \_stex_reset:N \l_stex_current_symbol_str
3068             \bool_set_true:N \l_stex_allow_semantic_bool
3069           }
3070           \def\comp{\_comp}
3071           \str_set:Nn \l_stex_current_symbol_str { #1 }
3072           \bool_set_false:N \l_stex_allow_semantic_bool
3073           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3074         }{
3075           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3076             ~\l_stex_notation_variant_str
3077           }
3078         }
3079       }{
3080         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3081       }
3082     }
3083   }{
3084     \_stex_reset:N \comp
3085     \_stex_reset:N \l_stex_current_symbol_str
3086     \bool_set_true:N \l_stex_allow_semantic_bool
3087   }
3088 }
3089
3090 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3091   \stex_find_notation:nn { #1 }{ #2 }
3092   \cs_if_exist:cTF {
3093     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3094   }{
3095     \tl_set:Nx \stex_symbol_after_invokation_tl {
3096       \_stex_reset:N \comp

```

```

3097     \stex_reset:N \stex_symbol_after_invokation_tl
3098     \stex_reset:N \l_stex_current_symbol_str
3099     \bool_set_true:N \l_stex_allow_semantic_bool
3100   }
3101   \def\comp{\_comp}
3102   \str_set:Nn \l_stex_current_symbol_str { #1 }
3103   \bool_set_false:N \l_stex_allow_semantic_bool
3104   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3105 }{
3106   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3107     ~\l_stex_notation_variant_str
3108   }
3109 }
3110 }
3111
3112 \prop_new:N \l__stex_terms_custom_args_prop
3113
3114 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3115   \exp_args:Nnx \use:nn {
3116     \bool_set_false:N \l_stex_allow_semantic_bool
3117     \def\comp{\_comp}
3118     \str_set:Nn \l_stex_current_symbol_str { #1 }
3119     \prop_clear:N \l__stex_terms_custom_args_prop
3120     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3121     \prop_get:cnN {
3122       l_stex_symdecl_#1 _prop
3123     }{ args } \l_tmpa_str
3124     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3125     \tl_set:Nn \arg { \__stex_terms_arg: }
3126     \str_if_empty:NTF \l_tmpa_str {
3127       \stex_term_oms:nnn {#1}{#1}{#2}
3128     }{
3129       \str_if_in:NnTF \l_tmpa_str b {
3130         \stex_term_ombind:nnn {#1}{#1}{#2}
3131       }{
3132         \str_if_in:NnTF \l_tmpa_str B {
3133           \stex_term_ombind:nnn {#1}{#1}{#2}
3134         }{
3135           \stex_term_oma:nnn {#1}{#1}{#2}
3136         }
3137       }
3138     }
3139     % TODO check that all arguments exist
3140   }{
3141     \stex_reset:N \l_stex_current_symbol_str
3142     \stex_reset:N \arg
3143     \stex_reset:N \comp
3144     \stex_reset:N \l__stex_terms_custom_args_prop
3145     \bool_set_true:N \l_stex_allow_semantic_bool
3146   }
3147 }
3148
3149 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3150   \tl_if_empty:nTF {#2}{

```

```

3151 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3152 \bool_set_true:N \l_tmpa_bool
3153 \bool_do_while:Nn \l_tmpa_bool {
3154   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3155   \int_incr:N \l_tmpa_int
3156   }{
3157     \bool_set_false:N \l_tmpa_bool
3158   }
3159 }
3160 }{
3161   \int_set:Nn \l_tmpa_int { #2 }
3162   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3163     % TODO throw error
3164   }
3165 }
3166 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3167 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3168   % TODO throw error
3169 }
3170 \bool_set_true:N \l_stex_allow_semantic_bool
3171 \IfBooleanTF#1{
3172   \stex_annotate_invisible:n {
3173     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3174   }
3175 }{
3176   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3177 }
3178 \bool_set_false:N \l_stex_allow_semantic_bool
3179 }
3180
3181
3182 \cs_new_protected:Nn \_stex_term_arg:nn {
3183   \bool_set_true:N \l_stex_allow_semantic_bool
3184   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3185   \bool_set_false:N \l_stex_allow_semantic_bool
3186 }
3187
3188 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3189   \exp_args:Nnx \use:nn
3190   { \int_set:Nn \l__stex_terms_downprec { #2 }
3191     \_stex_term_arg:nn { #1 }{ #3 }
3192   }
3193   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3194 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3195 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3196   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3197   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3198   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3199     \expandafter\if\expandafter\relax\noexpand#3
3200     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```

```

3201 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3202 }{
3203 \__stex_terms_math_assoc_arg_simple:n{#3}
3204 }
3205 }
3206
3207 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3208 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3209 \str_if_empty:NTF \l_tmpa_str {
3210 \exp_args:Nx \cs_if_eq:NNTF {
3211 \tl_head:N #1
3212 } \stex_invoke_sequence:n {
3213 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3214 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3215 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3216 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3217 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3218 \exp_not:n{\exp_args:Nnx \use:nn} {
3219 \exp_not:n {
3220 \def\comp{\_varcomp}
3221 \str_set:Nn \l_stex_current_symbol_str
3222 } {varseq://\l_tmpa_str}
3223 \exp_not:n{ ##1 }
3224 }{
3225 \exp_not:n {
3226 \_stex_reset:N \comp
3227 \_stex_reset:N \l_stex_current_symbol_str
3228 }
3229 }
3230 }}}
3231 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3232 \seq_reverse:N \l_tmpa_seq
3233 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3234 \seq_map_inline:Nn \l_tmpa_seq {
3235 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3236 \exp_args:Nno
3237 \l_tmpa_cs { ##1 } \l_tmpa_tl
3238 }
3239 }
3240 \tl_set:Nx \l_tmpa_tl {
3241 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3242 \exp_args:No \exp_not:n \l_tmpa_tl
3243 }
3244 }
3245 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3246 }{
3247 \__stex_terms_math_assoc_arg_simple:n { #1 }
3248 }
3249 } {
3250 \__stex_terms_math_assoc_arg_simple:n { #1 }
3251 }
3252
3253 }
3254

```

```

3255 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3256   \clist_set:Nn \l_tmpa_clist{ #1 }
3257   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3258     \tl_set:Nn \l_tmpa_tl { #1 }
3259   }{
3260     \clist_reverse:N \l_tmpa_clist
3261     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3262
3263     \clist_map_inline:Nn \l_tmpa_clist {
3264       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3265         \exp_args:Nno
3266         \l_tmpa_cs { ##1 } \l_tmpa_tl
3267       }
3268     }
3269   }
3270   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3271 }

```

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

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3272 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3273 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3274 \int_new:N \l__stex_terms_downprec
3275 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

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

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3276 \tl_set:Nn \l__stex_terms_left_bracket_str (
3277 \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

```

3278 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3279   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3280     \bool_set_false:N \l__stex_terms_brackets_done_bool
3281     #2
3282   } {
3283     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3284       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3285         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3286         \dobrackets { #2 }
3287       }
3288     }{ #2 }
3289   }
3290 }

```


(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

3291 \bool_new:N \l__stex_terms_brackets_done_bool
3292 %\RequirePackage{scalerel}
3293 \cs_new_protected:Npn \dobrackets #1 {
3294   %\ThisStyle{\if D\m@switch
3295   %   \exp_args:Nnx \use:nn
3296   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3297   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3298   % \else
3299   \exp_args:Nnx \use:nn
3300   {
3301     \bool_set_true:N \l__stex_terms_brackets_done_bool
3302     \int_set:Nn \l__stex_terms_downprec \infpref
3303     \l__stex_terms_left_bracket_str
3304     #1
3305   }
3306   {
3307     \bool_set_false:N \l__stex_terms_brackets_done_bool
3308     \l__stex_terms_right_bracket_str
3309     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3310   }
3311   %\fi}
3312 }
```

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

`\withbrackets`

```

3313 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3314   \exp_args:Nnx \use:nn
3315   {
3316     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3317     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3318     #3
3319   }
3320   {
3321     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3322     {\l__stex_terms_left_bracket_str}
3323     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3324     {\l__stex_terms_right_bracket_str}
3325   }
3326 }
```

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

`\STEXinvisible`

```

3327 \cs_new_protected:Npn \STEXinvisible #1 {
3328   \stex_annotate_invisible:n { #1 }
3329 }
```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
3330 \cs_new_protected:Nn \_stex_term_oms:nnn {
3331   \stex_annotate:nnn{ OMID }{ #2 }{
3332     \stex_highlight_term:nn { #1 } { #3 }
3333   }
3334 }
3335
3336 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3337   \__stex_terms_maybe_brackets:nn { #3 }{
3338     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3339   }
3340 }
```

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

`_stex_term_math_omv:nn`

```
3341 \cs_new_protected:Nn \_stex_term_omv:nn {
3342   \stex_annotate:nnn{ OMV }{ #1 }{
3343     \stex_highlight_term:nn { #1 } { #2 }
3344   }
3345 }
```

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

`_stex_term_math_oma:nnnn`

```
3346 \cs_new_protected:Nn \_stex_term_oma:nnn {
3347   \stex_annotate:nnn{ OMA }{ #2 }{
3348     \stex_highlight_term:nn { #1 } { #3 }
3349   }
3350 }
3351
3352 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3353   \__stex_terms_maybe_brackets:nn { #3 }{
3354     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3355   }
3356 }
```

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

`_stex_term_math_omb:nnnn`

```
3357 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3358   \stex_annotate:nnn{ OMBIND }{ #2 }{
3359     \stex_highlight_term:nn { #1 } { #3 }
3360   }
3361 }
3362
3363 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3364   \__stex_terms_maybe_brackets:nn { #3 }{
3365     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3366   }
3367 }
```

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

```

\symref
\symname
3368 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3369
3370 \keys_define:nn { stex / symname } {
3371   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3372   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3373   root     .tl_set_x:N = \l__stex_terms_root_tl
3374 }
3375
3376 \cs_new_protected:Nn \stex_symname_args:n {
3377   \tl_clear:N \l__stex_terms_post_tl
3378   \tl_clear:N \l__stex_terms_pre_tl
3379   \tl_clear:N \l__stex_terms_root_str
3380   \keys_set:nn { stex / symname } { #1 }
3381 }
3382
3383 \NewDocumentCommand \symref { m m }{
3384   \let\compemph_uri_prev:\compemph@uri
3385   \let\compemph@uri\symrefemph@uri
3386   \STEXsymbol{#1}!\{ #2 }
3387   \let\compemph@uri\compemph_uri_prev:
3388 }
3389
3390 \NewDocumentCommand \synonym { 0{} m m }{
3391   \stex_symname_args:n { #1 }
3392   \let\compemph_uri_prev:\compemph@uri
3393   \let\compemph@uri\symrefemph@uri
3394   % TODO
3395   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3396   \let\compemph@uri\compemph_uri_prev:
3397 }
3398
3399 \NewDocumentCommand \symname { 0{} m }{
3400   \stex_symname_args:n { #1 }
3401   \stex_get_symbol:n { #2 }
3402   \str_set:Nx \l_tmpa_str {
3403     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3404   }
3405   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3406
3407   \let\compemph_uri_prev:\compemph@uri
3408   \let\compemph@uri\symrefemph@uri
3409   \exp_args:NNx \use:nn
3410   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3411     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3412   } }
3413   \let\compemph@uri\compemph_uri_prev:
3414 }
3415
3416 \NewDocumentCommand \Symname { 0{} m }{
3417   \stex_symname_args:n { #1 }
3418   \stex_get_symbol:n { #2 }
3419   \str_set:Nx \l_tmpa_str {
3420     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3421 }
3422 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3423 \let\compemph_uri_prev:\compemph@uri
3424 \let\compemph@uri\symrefemph@uri
3425 \exp_args:NNx \use:nn
3426 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3427   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3428   \l__stex_terms_post_tl
3429 } }
3430 \let\compemph@uri\compemph_uri_prev:
3431 }

```

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

31.3 Notation Components

```

3432 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3433 \cs_new_protected:Nn \stex_highlight_term:nn {
3434   #2
3435 }
3436
3437 \cs_new_protected:Nn \stex_unhighlight_term:n {
3438   % \latexml_if:TF {
3439   %   #1
3440   % } {
3441   %   \rustex_if:TF {
3442   %     #1
3443   %   } {
3444   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3445   %   }
3446   % }
3447 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3448 \cs_new_protected:Npn \_comp #1 {
3449   \str_if_empty:NF \l_stex_current_symbol_str {
3450     \rustex_if:TF {
3451       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3452     }{
3453       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3454     }
3455   }
3456 }
3457
3458 \cs_new_protected:Npn \_varcomp #1 {
3459   \str_if_empty:NF \l_stex_current_symbol_str {
3460     \rustex_if:TF {
3461       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3462     }{
3463       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3464     }
3465   }
3466 }
3467
3468 \def\comp{\_comp}
3469
3470 \cs_new_protected:Npn \compemph@uri #1 #2 {
3471   \compemph{ #1 }
3472 }
3473
3474
3475 \cs_new_protected:Npn \compemph #1 {
3476   #1
3477 }
3478
3479 \cs_new_protected:Npn \defemph@uri #1 #2 {
3480   \defemph{#1}
3481 }
3482
3483 \cs_new_protected:Npn \defemph #1 {
3484   \textbf{#1}
3485 }
3486
3487 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3488   \symrefemph{#1}
3489 }
3490
3491 \cs_new_protected:Npn \symrefemph #1 {
3492   \textbf{#1}
3493 }
3494
3495 \cs_new_protected:Npn \varemp@uri #1 #2 {
3496   \varemp{#1}
3497 }
3498
3499 \cs_new_protected:Npn \varemp #1 {
3500   #1
3501 }

```

(End definition for `\comp` and others. These functions are documented on page 39.)

\ellipses

```

3502 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3503 \bool_new:N \l_stex_inparray_bool
3504 \bool_set_false:N \l_stex_inparray_bool
3505 \NewDocumentCommand \parray { m m } {
3506   \begingroup
3507   \bool_set_true:N \l_stex_inparray_bool
3508   \begin{array}{#1}
3509     #2
3510   \end{array}

```

```

3511 \endgroup
3512 }
3513
3514 \NewDocumentCommand \prmatrix { m } {
3515   \begingroup
3516   \bool_set_true:N \l_stex_inarray_bool
3517   \begin{matrix}
3518     #1
3519   \end{matrix}
3520 \endgroup
3521 }
3522
3523 \def \maybepline {
3524   \bool_if:NT \l_stex_inarray_bool {\hline}
3525 }
3526
3527 \def \parrayline #1 #2 {
3528   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3529 }
3530
3531 \def \pmrow #1 { \parrayline{}{ #1 } }
3532
3533 \def \parraylineh #1 #2 {
3534   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3535 }
3536
3537 \def \parraycell #1 {
3538   #1 \bool_if:NT \l_stex_inarray_bool {&}
3539 }

```

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

31.4 Variables

```

3540 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3541 \cs_new_protected:Nn \stex_invoke_variable:n {
3542   \if_mode_math:
3543     \exp_after:wN \__stex_variables_invoke_math:n
3544   \else:
3545     \exp_after:wN \__stex_variables_invoke_text:n
3546   \fi: {#1}
3547 }
3548
3549 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3550   %TODO
3551 }
3552
3553
3554 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3555   \peek_charcode_remove:NTF ! {
3556     \peek_charcode_remove:NTF ! {
3557       \peek_charcode:NTF [ {

```

```

3558     \__stex_variables_invoke_op_custom:nw
3559   }{
3560     % TODO throw error
3561   }
3562   ){
3563     \__stex_variables_invoke_op:n { #1 }
3564   }
3565   ){
3566     \peek_charcode_remove:NTF * {
3567       \__stex_variables_invoke_text:n { #1 }
3568     }{
3569       \__stex_variables_invoke_math_ii:n { #1 }
3570     }
3571   }
3572 }
3573
3574 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3575   \cs_if_exist:cTF {
3576     stex_var_op_notation_ #1 _cs
3577   }{
3578     \exp_args:Nnx \use:nn {
3579       \def\comp{\_varcomp}
3580       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3581       \_stex_term_omv:nn { var://#1 }{
3582         \use:c{stex_var_op_notation_ #1 _cs }
3583       }
3584     }{
3585       \_stex_reset:N \comp
3586       \_stex_reset:N \l_stex_current_symbol_str
3587     }
3588   }{
3589     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3590       \__stex_variables_invoke_math_ii:n {#1}
3591     }{
3592       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3593     }
3594   }
3595 }
3596
3597 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3598   \cs_if_exist:cTF {
3599     stex_var_notation_#1_cs
3600   }{
3601     \tl_set:Nx \stex_symbol_after_invokation_tl {
3602       \_stex_reset:N \comp
3603       \_stex_reset:N \stex_symbol_after_invokation_tl
3604       \_stex_reset:N \l_stex_current_symbol_str
3605       \bool_set_true:N \l_stex_allow_semantic_bool
3606     }
3607     \def\comp{\_varcomp}
3608     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3609     \bool_set_false:N \l_stex_allow_semantic_bool
3610     \use:c{stex_var_notation_#1_cs}
3611   }{

```

```

3612 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3613 }
3614 }

```

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

31.5 Sequences

```

3615 <@@=stex_sequences>
3616
3617 \cs_new_protected:Nn \stex_invoke_sequence:n {
3618   \peek_charcode_remove:NTF ! {
3619     \stex_term_omv:nn {varseq://#1}{
3620       \exp_args:Nnx \use:nn {
3621         \def\comp{\_varcomp}
3622         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3623         \prop_item:cn{stex_varseq_#1_prop}{notation}
3624       }{
3625         \stex_reset:N \comp
3626         \stex_reset:N \l_stex_current_symbol_str
3627       }
3628     }
3629   }{
3630     \bool_set_false:N \l_stex_allow_semantic_bool
3631     \def\comp{\_varcomp}
3632     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3633     \tl_set:Nx \stex_symbol_after_invokation_tl {
3634       \stex_reset:N \comp
3635       \stex_reset:N \stex_symbol_after_invokation_tl
3636       \stex_reset:N \l_stex_current_symbol_str
3637       \bool_set_true:N \l_stex_allow_semantic_bool
3638     }
3639     \use:c { stex_varseq_#1_cs }
3640   }
3641 }
3642 </package>

```


Chapter 32

STEX -Structural Features Implementation

```
3643 ⟨*package⟩
3644
3645 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3646
3647
3648 Warnings and error messages
3649 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3650   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3651 }
3652 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3653   Symbol~#1~not~assigned~in~interpretmodule~#2
3654 }
3655 \msg_new:nnn{stex}{error/unknownstructure}{
3656   No~structure~#1~found!
3657 }
3658 \msg_new:nnn{stex}{error/unknownfield}{
3659   No~field~#1~in~instance~#2~found!
3660 }
3661 \msg_new:nnn{stex}{error/keyval}{
3662   Invalid~key=value~pair~#1
3663 }
3664 \msg_new:nnn{stex}{error/instantiate/missing}{
3665   Assignments~missing~in~instantiate:~#1
3666 }
3667 \msg_new:nnn{stex}{error/incompatible}{
3668   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3669 }
3670
3671
```

32.1 Imports with modification

```

3672 <@@=stex_copymodule>
3673 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3674   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3675     \tl_set:Nn \l_tmpa_tl { #1 }
3676     \__stex_copymodule_get_symbol_from_cs:
3677   }{
3678     % argument is a string
3679     % is it a command name?
3680     \cs_if_exist:cTF { #1 }{
3681       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3682       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3683       \str_if_empty:NNTF \l_tmpa_str {
3684         \exp_args:Nx \cs_if_eq:NNTF {
3685           \tl_head:N \l_tmpa_tl
3686         } \stex_invoke_symbol:n {
3687           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3688         }{
3689           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3690         }
3691       } {
3692         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3693       }
3694     }{
3695       % argument is not a command name
3696       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3697       % \l_stex_all_symbols_seq
3698     }
3699   }
3700 }
3701
3702 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3703   \str_set:Nn \l_tmpa_str { #1 }
3704   \bool_set_false:N \l_tmpa_bool
3705   \bool_if:NF \l_tmpa_bool {
3706     \tl_set:Nn \l_tmpa_tl {
3707       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3708     }
3709     \str_set:Nn \l_tmpa_str { #1 }
3710     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3711     \seq_map_inline:Nn #2 {
3712       \str_set:Nn \l_tmpb_str { ##1 }
3713       \str_if_eq:eeT { \l_tmpa_str } {
3714         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3715       } {
3716         \seq_map_break:n {
3717           \tl_set:Nn \l_tmpa_tl {
3718             \str_set:Nn \l_stex_get_symbol_uri_str {
3719               ##1
3720             }
3721           }
3722         }
3723       }

```

```

3724     }
3725     \l_tmpa_tl
3726   }
3727 }
3728
3729 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3730   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3731     { \tl_tail:N \l_tmpa_tl }
3732   \tl_if_single:NTF \l_tmpa_tl {
3733     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3734       \exp_after:wN \str_set:Nn \exp_after:wN
3735         \l_stex_get_symbol_uri_str \l_tmpa_tl
3736       \__stex_copymodule_get_symbol_check:n { #1 }
3737     }{
3738       % TODO
3739       % tail is not a single group
3740     }
3741   }{
3742     % TODO
3743     % tail is not a single group
3744   }
3745 }
3746
3747 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3748   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3749     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3750       :~\seq_use:Nn #1 {,~}
3751     }
3752   }
3753 }
3754
3755 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3756   \stex_import_module_uri:nn { #1 } { #2 }
3757   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3758   \stex_import_require_module:nnnn
3759     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3760     { \l_stex_import_path_str } { \l_stex_import_name_str }
3761   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3762   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3763   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3764   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3765     \seq_map_inline:cn {c_stex_module_###1_constants}{
3766       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3767         ##1 ? ####1
3768       }
3769     }
3770   }
3771   \seq_clear:N \l_tmpa_seq
3772   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3773     name      = \l_stex_current_copymodule_name_str ,
3774     module    = \l_stex_current_module_str ,
3775     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3776     includes  = \l_tmpa_seq ,
3777     fields    = \l_tmpa_seq

```

```

3778 }
3779 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3780   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3781   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3782 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3783 \stex_if_smsmode:F {
3784   \begin{stex_annotate_env} {#4} {
3785     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3786   }
3787   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3788 }
3789 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3790 \bool_set_false:N \_stex_html_do_output_bool
3791 }
3792 \cs_new_protected:Nn \stex_copymodule_end:n {
3793   \def \l_tmpa_cs ##1 ##2 {#1}
3794   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3795   \tl_clear:N \l_tmpa_tl
3796   \tl_clear:N \l_tmpb_tl
3797   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3798   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3799     \seq_map_inline:cn {c_stex_module_##1_constants}{
3800       \tl_clear:N \l_tmpc_tl
3801       \l_tmpa_cs{##1}{####1}
3802       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3803         \tl_put_right:Nx \l_tmpa_tl {
3804           \prop_set_from_keyval:cn {
3805             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3806           }{
3807             \exp_after:wN \prop_to_keyval:N \csname
3808               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3809             \endcsname
3810           }
3811           \seq_clear:c {
3812             l_stex_symdecl_
3813             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
3814             _notations
3815           }
3816         }
3817         \tl_put_right:Nx \l_tmpc_tl {
3818           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
3819           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1
3820         }
3821         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
3822         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3823           \tl_put_right:Nx \l_tmpc_tl {
3824             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3825           }
3826           \tl_put_right:Nx \l_tmpa_tl {
3827             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3828               \stex_invoke_symbol:n {
3829                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
3830             }
3831           }

```

```

3832     }
3833   }
3834 }{
3835   \tl_put_right:Nx \l_tmpc_tl {
3836     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3837   }
3838   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3839   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3840   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3841   \tl_put_right:Nx \l_tmpa_tl {
3842     \prop_set_from_keyval:cn {
3843       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3844     }{
3845       \prop_to_keyval:N \l_tmpa_prop
3846     }
3847     \seq_clear:c {
3848       l_stex_symdecl_
3849       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3850       _notations
3851     }
3852   }
3853   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3854   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3855     \tl_put_right:Nx \l_tmpc_tl {
3856       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3857     }
3858     \tl_put_right:Nx \l_tmpa_tl {
3859       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3860         \stex_invoke_symbol:n {
3861           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3862         }
3863       }
3864     }
3865   }
3866 }
3867 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3868   \tl_put_right:Nx \l_tmpc_tl {
3869     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3870   }
3871 }
3872 \tl_put_right:Nx \l_tmpb_tl {
3873   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3874 }
3875 }
3876 }
3877 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3878 \tl_put_left:Nx \l_tmpa_tl {
3879   \prop_set_from_keyval:cn {
3880     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3881   }{
3882     \prop_to_keyval:N \l_stex_current_copymodule_prop
3883   }
3884 }
3885 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3886 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3887 \exp_args:Nx \stex_do_up_to_module:n {
3888   \exp_args:No \exp_not:n \l_tmpa_tl
3889 }
3890 \l_tmpb_tl
3891 \stex_if_smsmode:F {
3892   \end{stex_annotate_env}
3893 }
3894 }
3895
3896 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3897   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3898   \stex_deactivate_macro:Nn \symdecl {module~environments}
3899   \stex_deactivate_macro:Nn \symdef {module~environments}
3900   \stex_deactivate_macro:Nn \notation {module~environments}
3901   \stex_reactivate_macro:N \assign
3902   \stex_reactivate_macro:N \renamedec1
3903   \stex_reactivate_macro:N \donotcopy
3904   \stex_smsmode_do:
3905 }{
3906   \stex_copymodule_end:n {}
3907 }
3908
3909 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3910   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3911   \stex_deactivate_macro:Nn \symdecl {module~environments}
3912   \stex_deactivate_macro:Nn \symdef {module~environments}
3913   \stex_deactivate_macro:Nn \notation {module~environments}
3914   \stex_reactivate_macro:N \assign
3915   \stex_reactivate_macro:N \renamedec1
3916   \stex_reactivate_macro:N \donotcopy
3917   \stex_smsmode_do:
3918 }{
3919   \stex_copymodule_end:n {
3920     \tl_if_exist:cF {
3921       l__stex_copymodule_copymodule_##1?##2_def_tl
3922     }{
3923       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3924         ##1?##2
3925       }{\l_stex_current_copymodule_name_str}
3926     }
3927   }
3928 }
3929
3930 \NewDocumentCommand \donotcopy { 0{} m}{
3931   \stex_import_module_uri:nn { #1 } { #2 }
3932   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3933   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3934     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
3935     \seq_map_inline:cn {c_stex_module_##1_constants}{
3936       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3937       \bool_lazy_any_p:nT {
3938         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3939         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}

```

```

3940         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3941     }{
3942         % TODO throw error
3943     }
3944 }
3945 }
3946
3947 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3948 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3949 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3950 }
3951
3952 \NewDocumentCommand \assign { m m }{
3953     \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3954     \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3955     \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3956 }
3957
3958 \keys_define:nn { stex / renamedec1 } {
3959     name .str_set_x:N = \l_stex_renamedec1_name_str
3960 }
3961 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
3962     \str_clear:N \l_stex_renamedec1_name_str
3963     \keys_set:nn { stex / renamedec1 } { #1 }
3964 }
3965
3966 \NewDocumentCommand \renamedec1 { O{} m m }{
3967     \__stex_copymodule_renamedec1_args:n { #1 }
3968     \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
3969     \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3970     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3971     \str_if_empty:NTF \l_stex_renamedec1_name_str {
3972         \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3973             \l_stex_get_symbol_uri_str
3974         } }
3975     } {
3976         \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
3977         \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3978         \prop_set_eq:cc {l_stex_symdecl_
3979             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3980             _prop
3981         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3982         \seq_set_eq:cc {l_stex_symdecl_
3983             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3984             _notations
3985         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3986         \prop_put:cnx {l_stex_symdecl_
3987             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3988             _prop
3989         }{ name }{ \l_stex_renamedec1_name_str }
3990         \prop_put:cnx {l_stex_symdecl_
3991             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3992             _prop
3993         }{ module }{ \l_stex_current_module_str }

```

```

3994 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
3995 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3996 }
3997 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3998 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3999 } }
4000 }
4001 }
4002
4003 \stex_deactivate_macro:Nn \assign {copymodules}
4004 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4005 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4006
4007
4008 \seq_new:N \l_stex_implicit_morphisms_seq
4009 \NewDocumentCommand \implicitmorphism { 0{} m m}{
4010 \stex_import_module_uri:nn { #1 } { #2 }
4011 \stex_debug:nn{implicits}{
4012 Implicit~morphism:~
4013 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4014 }
4015 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4016 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4017 }{
4018 \msg_error:nnn{stex}{error/conflictingmodules}{
4019 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4020 }
4021 }
4022
4023 % TODO
4024
4025
4026
4027 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4028 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4029 }
4030 }
4031

```

32.2 The feature environment

structural@feature

```

4032 <@@=stex_features>
4033
4034 \NewDocumentEnvironment{structural_feature_module}{ m m m m }{
4035 \stex_if_in_module:F {
4036 \msg_set:nnn{stex}{error/nomodule}{
4037 Structural~Feature~has~to~occur~in~a~module:\\
4038 Feature~#2~of~type~#1\\
4039 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4040 }
4041 \msg_error:nn{stex}{error/nomodule}
4042 }

```



```

4043
4044 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4045
4046 \stex_if_smsmode:F {
4047   \begin{stex_annotate_env}{ feature:#1 }{}
4048   \stex_annotate_invisible:nnn{header}{}{ #3 }
4049 }
4050 }{
4051   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4052   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4053   \stex_debug:nn{features}{
4054     Feature: \l_stex_last_feature_str
4055   }
4056   \stex_if_smsmode:F {
4057     \end{stex_annotate_env}
4058   }
4059 }

```

32.3 Structure

structure

```

4060 <@@=stex_structures>
4061 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4062   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4063     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4064   }
4065   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4066   {#1}{#2}
4067 }
4068
4069 \keys_define:nn { stex / features / structure } {
4070   name .str_set_x:N = \l__stex_structures_name_str ,
4071 }
4072
4073 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4074   \str_clear:N \l__stex_structures_name_str
4075   \keys_set:nn { stex / features / structure } { #1 }
4076 }
4077
4078 \NewDocumentEnvironment{mathstructure}{m O{}}{
4079   \__stex_structures_structure_args:n { #2 }
4080   \str_if_empty:NT \l__stex_structures_name_str {
4081     \str_set:Nx \l__stex_structures_name_str { #1 }
4082   }
4083   \exp_args:Nnnx
4084   \begin{structural_feature_module}{ structure }
4085     { \l__stex_structures_name_str }{}
4086   \stex_smsmode_do:
4087 }{
4088   \end{structural_feature_module}
4089   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4090   \seq_clear:N \l_tmpa_seq
4091   \seq_map_inline:Nn \l_stex_collect_imports_seq {

```

```

4092     \seq_map_inline:cn{c_stex_module_##1_constants}{
4093       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4094     }
4095   }
4096   \exp_args:Nnno
4097   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4098   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4099   \stex_add_structure_to_current_module:nn
4100     \l__stex_structures_name_str
4101     \l_stex_last_feature_str
4102   \exp_args:Nx \stex_symdecl_do:nn {
4103     name = \l__stex_structures_name_str ,
4104     type = \metacollection ,
4105     def = {\STEXsymbol{module-type}{
4106       \stex_term_math_oms:nnnn { \l_stex_last_feature_str }{}{0}{}}
4107     }}
4108   }{ #1 }
4109   \exp_args:Nx
4110   \stex_add_to_current_module:n {
4111     \tl_set:cn { #1 }{
4112       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4113     }
4114   }
4115   \exp_args:Nx
4116   \stex_do_up_to_module:n {
4117     \tl_set:cn { #1 }{
4118       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4119     }
4120   }
4121 }
4122 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4123
4124 \cs_new:Nn \stex_invoke_structure:nn {
4125   \stex_invoke_symbol:n { #1?#2 }
4126 }
4127
4128 \cs_new_protected:Nn \stex_get_structure:n {
4129   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4130     \tl_set:Nn \l_tmpa_tl { #1 }
4131     \__stex_structures_get_from_cs:
4132   }{
4133     \cs_if_exist:cTF { #1 }{
4134       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4135       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4136       \str_if_empty:NNTF \l_tmpa_str {
4137         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4138           \__stex_structures_get_from_cs:
4139         }{
4140           \__stex_structures_get_from_string:n { #1 }
4141         }
4142       }{
4143         \__stex_structures_get_from_string:n { #1 }
4144       }
4145     }{

```

```

4146     \__stex_structures_get_from_string:n { #1 }
4147   }
4148 }
4149 }
4150
4151 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4152   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4153     { \tl_tail:N \l_tmpa_tl }
4154   \str_set:Nx \l_tmpa_str {
4155     \exp_after:wN \use_i:nn \l_tmpa_tl
4156   }
4157   \str_set:Nx \l_tmpb_str {
4158     \exp_after:wN \use_ii:nn \l_tmpa_tl
4159   }
4160   \str_set:Nx \l_stex_get_structure_str {
4161     \l_tmpa_str ? \l_tmpb_str
4162   }
4163   \str_set:Nx \l_stex_get_structure_module_str {
4164     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4165   }
4166 }
4167
4168 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4169   \tl_set:Nn \l_tmpa_tl {
4170     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4171   }
4172   \str_set:Nn \l_tmpa_str { #1 }
4173   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4174
4175   \seq_map_inline:Nn \l_stex_all_modules_seq {
4176     \prop_if_exist:cT {c_stex_module_##1_structures} {
4177       \prop_map_inline:cn {c_stex_module_##1_structures} {
4178         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4179           \prop_map_break:n{\seq_map_break:n{
4180             \tl_set:Nn \l_tmpa_tl {
4181               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4182               \str_set:Nn \l_stex_get_structure_module_str {####2}
4183             }
4184           }}
4185         }
4186       }
4187     }
4188   }
4189   \l_tmpa_tl
4190 }

```

\instantiate

```

4191
4192 \keys_define:nn { stex / instantiate } {
4193   name .str_set_x:N = \l__stex_structures_name_str
4194 }
4195 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4196   \str_clear:N \l__stex_structures_name_str
4197   \keys_set:nn { stex / instantiate } { #1 }

```

```

4198 }
4199
4200 \NewDocumentCommand \instantiate {m O{} m m m}{
4201   \beginingroup
4202     \stex_get_structure:n {#4}
4203     \__stex_structures_instantiate_args:n { #2 }
4204     \str_if_empty:NT \l__stex_structures_name_str {
4205       \str_set:Nn \l__stex_structures_name_str { #1 }
4206     }
4207     \seq_clear:N \l__stex_structures_fields_seq
4208     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4209     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4210       \seq_map_inline:cn {c_stex_module_##1_constants}{
4211         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4212       }
4213     }
4214     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4215     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4216     \prop_clear:N \l_tmpa_prop
4217     \seq_map_inline:Nn \l_tmpa_seq {
4218       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4219       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4220         \msg_error:nnn{stex}{error/keyval}{##1}
4221       }
4222       \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4223       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4224       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4225       \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4226       \exp_args:Nxx \str_if_eq:nnF
4227         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4228         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4229         \msg_error:nnxxx{stex}{error/incompatible}
4230         {\l__stex_structures_dom_str}
4231         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4232         {\l_stex_get_symbol_uri_str}
4233         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4234       }
4235       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4236     }
4237     \seq_if_empty:NF \l__stex_structures_fields_seq {
4238       \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
4239     }
4240     \exp_args:Nx
4241     \stex_add_to_current_module:n {
4242       \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4243       domain = \l_stex_get_structure_module_str ,
4244       \prop_to_keyval:N \l_tmpa_prop
4245     }
4246     \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4247   }
4248   \exp_args:Nx
4249   \stex_do_up_to_module:n {
4250     \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4251     domain = \l_stex_get_structure_module_str ,

```

```

4252         \prop_to_keyval:N \l_tmpa_prop
4253     }
4254     \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structures_name_str}{\exp_not:n{\comp{#5}}}}
4255     \notation{\l__stex_structures_name_str}{\exp_not:n{\comp{#5}}}}
4256 }
4257 \exp_args:Nxx \stex_symdecl_do:nn {
4258     type={\STEXsymbol{module-type}}{
4259         \stex_term_math_oms:nnnn {
4260             \l_stex_get_structure_module_str
4261         }{0}{0}}
4262     }}
4263     }{\l__stex_structures_name_str}
4264 \endgroup
4265 \stex_smsmode_do:\ignorespacesandpars
4266 }
4267 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4268
4269 \cs_new_protected:Nn \stex_symbol_or_var:n {
4270     \cs_if_exist:cTF{#1}{
4271         \cs_set_eq:Nc \l_tmpa_tl { #1 }
4272         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4273         \str_if_empty:NTF \l_tmpa_str {
4274             \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4275             \stex_invoke_variable:n {
4276                 \bool_set_true:N \l_stex_symbol_or_var_bool
4277                 \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4278                 \str_set:Nx \l_stex_get_symbol_uri_str {
4279                     \exp_after:wN \use:n \l_tmpa_tl
4280                 }
4281             }{
4282                 \bool_set_false:N \l_stex_symbol_or_var_bool
4283                 \stex_get_symbol:n{#1}
4284             }
4285         }{
4286             \__stex_structures_symbolorvar_from_string:n{ #1 }
4287         }
4288     }{
4289         \__stex_structures_symbolorvar_from_string:n{ #1 }
4290     }
4291 }
4292
4293 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4294     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4295         \bool_set_true:N \l_stex_symbol_or_var_bool
4296         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4297     }{
4298         \bool_set_false:N \l_stex_symbol_or_var_bool
4299         \stex_get_symbol:n{#1}
4300     }
4301 }
4302
4303
4304 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4305     \begingroup

```

```

4306 \stex_get_structure:n {#4}
4307 \__stex_structures_instantiate_args:n { #2 }
4308 \str_if_empty:NT \l__stex_structures_name_str {
4309   \str_set:Nn \l__stex_structures_name_str { #1 }
4310 }
4311 \seq_clear:N \l__stex_structures_fields_seq
4312 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4313 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4314   \seq_map_inline:cn {c_stex_module_##1_constants}{
4315     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4316   }
4317 }
4318 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4319 \prop_clear:N \l_tmpa_prop
4320 \tl_if_empty:nF {#3} {
4321   \seq_set_split:Nnn \l_tmpa_seq , {#3}
4322   \seq_map_inline:Nn \l_tmpa_seq {
4323     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4324     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4325       \msg_error:nnn{stex}{error/keyval}{##1}
4326     }
4327     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_fields_seq
4328     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4329     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
4330     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4331     \bool_if:NTF \l_stex_symbol_or_var_bool {
4332       \exp_args:Nxx \str_if_eq:nnF
4333         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4334         {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4335         \msg_error:nnxxx{stex}{error/incompatible}
4336         {\l__stex_structures_dom_str
4337           {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4338           {\l_stex_get_symbol_uri_str
4339             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}}
4340       }
4341       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {\l__stex_structures_dom_str _prop}}
4342     }{
4343       \exp_args:Nxx \str_if_eq:nnF
4344         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4345         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4346         \msg_error:nnxxx{stex}{error/incompatible}
4347         {\l__stex_structures_dom_str
4348           {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4349           {\l_stex_get_symbol_uri_str
4350             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}}
4351       }
4352       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l__stex_structures_dom_str _prop}}
4353     }
4354   }
4355 }
4356 \tl_gclear:N \g__stex_structures_aftergroup_tl
4357 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4358   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq_##1_constants}{}}
4359   \stex_find_notation:nn{##1}{}}

```

```

4360 \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4361 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4362 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4363 \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4364 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4365 }
4366
4367 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4368 \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4369 name = \l_tmpa_str ,
4370 args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4371 arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4372 assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4373 }
4374 \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4375 {g__stex_structures_tmpa_\l_tmpa_str_cs}
4376 \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4377 {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4378 }
4379 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invoke
4380 }
4381 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4382 \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4383 domain = \l_stex_get_structure_module_str ,
4384 \prop_to_keyval:N \l_tmpa_prop
4385 }
4386 \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4387 \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4388 \exp_args:Nnx \exp_not:N \use:nn {
4389 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name_
4390 \stex_term_omv:nn {var://\l__stex_structures_name_str}{
4391 \exp_not:n{
4392 \varcomp{#5}
4393 }
4394 }
4395 }{
4396 \exp_not:n{\stex_reset:N \l_stex_current_symbol_str}
4397 }
4398 }
4399 }
4400 \aftergroup\g__stex_structures_aftergroup_tl
4401 \endgroup
4402 \stex_smsmode_do:\ignorespacesandpars
4403 }
4404
4405 \cs_new_protected:Nn \stex_invoke_instance:n {
4406 \peek_charcode_remove:NTF ! {
4407 \STEXsymbol{?#1}
4408 }{
4409 \stex_invoke_instance:nn {#1}
4410 }
4411 }
4412
4413

```

```

4414 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4415   \peek_charcode_remove:NTF ! {
4416     \use:c{l_stex_varinstance_#1_op_tl}
4417   }{
4418     \stex_invoke_varinstance:nn {#1}
4419   }
4420 }
4421
4422 \cs_new_protected:Nn \stex_invoke_instance:nn {
4423   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4424     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4425   }{
4426     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4427   }
4428 }
4429
4430 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4431   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4432     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4433     \l_tmpa_tl
4434   }{
4435     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4436   }
4437 }

```

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

\stex_invoke_structure:nnn

```

4438 % #1: URI of the instance
4439 % #2: URI of the instantiated module
4440 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4441   \tl_if_empty:nTF{ #3 }{
4442     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4443       c_stex_feature_ #2 _prop
4444     }
4445     \tl_clear:N \l_tmpa_tl
4446     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4447     \seq_map_inline:Nn \l_tmpa_seq {
4448       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4449       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4450       \cs_if_exist:cT {
4451         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4452       }{
4453         \tl_if_empty:NF \l_tmpa_tl {
4454           \tl_put_right:Nn \l_tmpa_tl {,}
4455         }
4456         \tl_put_right:Nx \l_tmpa_tl {
4457           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4458         }
4459       }
4460     }
4461     \exp_args:No \mathstrut \l_tmpa_tl
4462   }{
4463     \stex_invoke_symbol:n{#1/#3}

```



```

4464 }
4465 }

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

4466 </package>

```

Chapter 33

STEX -Statements Implementation

```
4467 <*package>
4468
4469 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4470
4471 <@@=stex_statements>
    Warnings and error messages
4472

\titleemph
4473 \def\titleemph#1{\textbf{#1}}

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

33.1 Definitions

```
definiendum

4474 \keys_define:nn {stex / definiendum }{
4475   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4476   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4477   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4478   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4479 }
4480 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4481   \str_clear:N \l__stex_statements_definiendum_root_str
4482   \tl_clear:N \l__stex_statements_definiendum_post_tl
4483   \str_clear:N \l__stex_statements_definiendum_gfa_str
4484   \keys_set:nn { stex / definiendum }{ #1 }
4485 }
4486 \NewDocumentCommand \definiendum { O{} m m } {
4487   \__stex_statements_definiendum_args:n { #1 }
4488   \stex_get_symbol:n { #2 }
4489   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4490   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4491     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4492     \tl_set:Nn \l_tmpa_tl { #3 }
4493   } {
4494     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4495     \tl_set:Nn \l_tmpa_tl {
4496       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4497     }
4498   }
4499 } {
4500   \tl_set:Nn \l_tmpa_tl { #3 }
4501 }
4502
4503 % TODO root
4504 \rustex_if:TF {
4505   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4506 } {
4507   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4508 }
4509 }
4510 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4511
4512 \NewDocumentCommand \definame { 0{ } m } {
4513   \__stex_statements_definiendum_args:n { #1 }
4514   % TODO: root
4515   \stex_get_symbol:n { #2 }
4516   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4517   \str_set:Nx \l_tmpa_str {
4518     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4519   }
4520   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4521   \rustex_if:TF {
4522     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4523       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4524     }
4525   } {
4526     \exp_args:Nnx \defemph@uri {
4527       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4528     } { \l_stex_get_symbol_uri_str }
4529   }
4530 }
4531 \stex_deactivate_macro:Nn \definame {definition~environments}
4532
4533 \NewDocumentCommand \Definame { 0{ } m } {
4534   \__stex_statements_definiendum_args:n { #1 }
4535   \stex_get_symbol:n { #2 }
4536   \str_set:Nx \l_tmpa_str {
4537     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4538   }
4539   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4540   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4541   \rustex_if:TF {

```

```

4542 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4543 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4544 }
4545 } {
4546 \exp_args:Nnx \defemph@uri {
4547 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4548 } { \l_stex_get_symbol_uri_str }
4549 }
4550 }
4551 \stex_deactivate_macro:Nn \Definame {definition~environments}
4552
4553 \NewDocumentCommand \premise { m }{
4554 \stex_annotate:nnn{ premise }{}{ #1 }
4555 }
4556 \NewDocumentCommand \conclusion { m }{
4557 \stex_annotate:nnn{ conclusion }{}{ #1 }
4558 }
4559 \NewDocumentCommand \definiens { m }{
4560 \stex_annotate:nnn{ definiens }{}{ #1 }
4561 }
4562
4563 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4564 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4565 \stex_deactivate_macro:Nn \definiens {definition~environments}
4566

```

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

sdefinition

```

4567
4568 \keys_define:nn {stex / sdefinition }{
4569 type .str_set_x:N = \sdefinitiontype,
4570 id .str_set_x:N = \sdefinitionid,
4571 name .str_set_x:N = \sdefinitionname,
4572 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4573 title .tl_set:N = \sdefinitiontitle
4574 }
4575 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4576 \str_clear:N \sdefinitiontype
4577 \str_clear:N \sdefinitionid
4578 \str_clear:N \sdefinitionname
4579 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4580 \tl_clear:N \sdefinitiontitle
4581 \keys_set:nn { stex / sdefinition }{}{ #1 }
4582 }
4583
4584 \NewDocumentEnvironment{sdefinition}{0{}}{
4585 \__stex_statements_sdefinition_args:n{ #1 }
4586 \stex_reactivate_macro:N \definiendum
4587 \stex_reactivate_macro:N \definame
4588 \stex_reactivate_macro:N \Definame
4589 \stex_reactivate_macro:N \premise
4590 \stex_reactivate_macro:N \definiens
4591 \stex_if_smsmode:F{

```

```

4592 \seq_clear:N \l_tmpa_seq
4593 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4594   \tl_if_empty:nF{ ##1 }{
4595     \stex_get_symbol:n { ##1 }
4596     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4597       \l_stex_get_symbol_uri_str
4598     }
4599   }
4600 }
4601 \exp_args:Nnnx
4602 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4603 \str_if_empty:NF \sdefinitiontype {
4604   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4605 }
4606 \clist_set:No \l_tmpa_clist \sdefinitiontype
4607 \tl_clear:N \l_tmpa_tl
4608 \clist_map_inline:Nn \l_tmpa_clist {
4609   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4610     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4611   }
4612 }
4613 \tl_if_empty:NTF \l_tmpa_tl {
4614   \__stex_statements_sdefinition_start:
4615 }{
4616   \l_tmpa_tl
4617 }
4618 }
4619 \stex_ref_new_doc_target:n \sdefinitionid
4620 \stex_smsmode_do:
4621 }{
4622   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4623   \stex_if_smsmode:F {
4624     \clist_set:No \l_tmpa_clist \sdefinitiontype
4625     \tl_clear:N \l_tmpa_tl
4626     \clist_map_inline:Nn \l_tmpa_clist {
4627       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4628         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4629       }
4630     }
4631     \tl_if_empty:NTF \l_tmpa_tl {
4632       \__stex_statements_sdefinition_end:
4633     }{
4634       \l_tmpa_tl
4635     }
4636     \end{stex_annotate_env}
4637   }
4638 }

```

\stexpatchdefinition

```

4639 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4640   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4641     ~(\sdefinitiontitle)
4642   }~}
4643 }

```

```

4644 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4645
4646 \newcommand\stexpatchdefinition[3] [] {
4647   \str_set:Nx \l_tmpa_str{ #1 }
4648   \str_if_empty:NTF \l_tmpa_str {
4649     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4650     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4651   }{
4652     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4653     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4654   }
4655 }

```

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

\inlinedef inline:

```

4656 \keys_define:nn {stex / inlinedef }{
4657   type      .str_set_x:N = \sdefinitiontype,
4658   id        .str_set_x:N = \sdefinitionid,
4659   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4660   name      .str_set_x:N = \sdefinitionname
4661 }
4662 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4663   \str_clear:N \sdefinitiontype
4664   \str_clear:N \sdefinitionid
4665   \str_clear:N \sdefinitionname
4666   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4667   \keys_set:nn { stex / inlinedef }{ #1 }
4668 }
4669 \NewDocumentCommand \inlinedef { 0{} m } {
4670   \beginingroup
4671   \__stex_statements_inlinedef_args:n{ #1 }
4672   \stex_reactivate_macro:N \definiendum
4673   \stex_reactivate_macro:N \definame
4674   \stex_reactivate_macro:N \Definame
4675   \stex_reactivate_macro:N \premise
4676   \stex_reactivate_macro:N \definiens
4677   \stex_ref_new_doc_target:n \sdefinitionid
4678   \stex_if_smsmode:TF{
4679     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4680   }{
4681     \seq_clear:N \l_tmpa_seq
4682     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4683       \tl_if_empty:nF{ ##1 }{
4684         \stex_get_symbol:n { ##1 }
4685         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4686           \l_stex_get_symbol_uri_str
4687         }
4688       }
4689     }
4690     \exp_args:Nnx
4691     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4692       \str_if_empty:NF \sdefinitiontype {
4693         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```

```

4694     }
4695     #2
4696     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4697   }
4698 }
4699 \endgroup
4700 \stex_smsmode_do:
4701 }

```

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

33.2 Assertions

sassertion

```

4702
4703 \keys_define:nn {stex / sassertion }{
4704   type      .str_set_x:N = \sassertiontype,
4705   id        .str_set_x:N = \sassertionid,
4706   title     .tl_set:N    = \sassertiontitle ,
4707   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4708   name      .str_set_x:N = \sassertionname
4709 }
4710 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4711   \str_clear:N \sassertiontype
4712   \str_clear:N \sassertionid
4713   \str_clear:N \sassertionname
4714   \clist_clear:N \l__stex_statements_sassertion_for_clist
4715   \tl_clear:N \sassertiontitle
4716   \keys_set:nn { stex / sassertion }{ #1 }
4717 }
4718
4719 %\tl_new:N \g__stex_statements_aftergroup_tl
4720
4721 \NewDocumentEnvironment{sassertion}{0{}}{
4722   \__stex_statements_sassertion_args:n{ #1 }
4723   \stex_reactivate_macro:N \premise
4724   \stex_reactivate_macro:N \conclusion
4725   \stex_if_smsmode:F {
4726     \seq_clear:N \l_tmpa_seq
4727     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4728       \tl_if_empty:nF{ ##1 }{
4729         \stex_get_symbol:n { ##1 }
4730         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4731           \l_stex_get_symbol_uri_str
4732         }
4733       }
4734     }
4735     \exp_args:Nnnx
4736     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4737     \str_if_empty:NF \sassertiontype {
4738       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4739     }
4740     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4741 \tl_clear:N \l_tmpa_tl
4742 \clist_map_inline:Nn \l_tmpa_clist {
4743   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4744     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4745   }
4746 }
4747 \tl_if_empty:NTF \l_tmpa_tl {
4748   \__stex_statements_sassertion_start:
4749 }{
4750   \l_tmpa_tl
4751 }
4752 }
4753 \str_if_empty:NTF \sassertionid {
4754   \str_if_empty:NF \sassertionname {
4755     \stex_ref_new_doc_target:n {}
4756   }
4757 } {
4758   \stex_ref_new_doc_target:n \sassertionid
4759 }
4760 \stex_smsmode_do:
4761 ){
4762   \str_if_empty:NF \sassertionname {
4763     \stex_symdecl_do:nn{ }\sassertionname}
4764     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4765   }
4766   \stex_if_smsmode:F {
4767     \clist_set:Nn \l_tmpa_clist \sassertiontype
4768     \tl_clear:N \l_tmpa_tl
4769     \clist_map_inline:Nn \l_tmpa_clist {
4770       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4771         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4772       }
4773     }
4774     \tl_if_empty:NTF \l_tmpa_tl {
4775       \__stex_statements_sassertion_end:
4776     }{
4777       \l_tmpa_tl
4778     }
4779     \end{stex_annotate_env}
4780   }
4781 }

```

\stexpatchassertion

```

4782
4783 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4784   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4785     (\sassertiontitle)
4786   }~}
4787 }
4788 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4789
4790 \newcommand\stexpatchassertion[3] [] {
4791   \str_set:Nx \l_tmpa_str{ #1 }
4792   \str_if_empty:NTF \l_tmpa_str {

```



```

4793     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4794     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4795   }{
4796     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4797     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4798   }
4799 }

```

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

\inlineass inline:

```

4800 \keys_define:nn {stex / inlineass }{
4801   type      .str_set_x:N = \sassertiontype,
4802   id        .str_set_x:N = \sassertionid,
4803   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4804   name      .str_set_x:N = \sassertionname
4805 }
4806 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4807   \str_clear:N \sassertiontype
4808   \str_clear:N \sassertionid
4809   \str_clear:N \sassertionname
4810   \clist_clear:N \l__stex_statements_sassertion_for_clist
4811   \keys_set:nn { stex / inlineass }{ #1 }
4812 }
4813 \NewDocumentCommand \inlineass { 0{} m } {
4814   \begin_group
4815   \stex_reactivate_macro:N \premise
4816   \stex_reactivate_macro:N \conclusion
4817   \__stex_statements_inlineass_args:n{ #1 }
4818   \str_if_empty:NTF \sassertionid {
4819     \str_if_empty:NF \sassertionname {
4820       \stex_ref_new_doc_target:n { }
4821     }
4822   } {
4823     \stex_ref_new_doc_target:n \sassertionid
4824   }
4825
4826   \stex_if_smsmode:TF{
4827     \str_if_empty:NF \sassertionname {
4828       \stex_symdecl_do:nn{}{\sassertionname}
4829       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4830     }
4831   }{
4832     \seq_clear:N \l_tmpa_seq
4833     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4834       \tl_if_empty:nF{ ##1 }{
4835         \stex_get_symbol:n { ##1 }
4836         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4837           \l_stex_get_symbol_uri_str
4838         }
4839       }
4840     }
4841     \exp_args:Nnx
4842     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4843 \str_if_empty:NF \sassertiontype {
4844 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4845 }
4846 #2
4847 \str_if_empty:NF \sassertionname {
4848 \stex_symdecl_do:nn{ }\sassertionname}
4849 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4850 }
4851 }
4852 }
4853 \endgroup
4854 \stex_smsmode_do:
4855 }

```

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

33.3 Examples

`sexample`

```

4856
4857 \keys_define:nn {stex / sexample }{
4858 type .str_set_x:N = \exampletype,
4859 id .str_set_x:N = \sexampleid,
4860 title .tl_set:N = \sexampletitle,
4861 for .clist_set:N = \l__stex_statements_sexample_for_clist,
4862 }
4863 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4864 \str_clear:N \sexampletype
4865 \str_clear:N \sexampleid
4866 \tl_clear:N \sexampletitle
4867 \clist_clear:N \l__stex_statements_sexample_for_clist
4868 \keys_set:nn { stex / sexample }{ #1 }
4869 }
4870
4871 \NewDocumentEnvironment{sexample}{0{}}{
4872 \__stex_statements_sexample_args:n{ #1 }
4873 \stex_reactivate_macro:N \premise
4874 \stex_reactivate_macro:N \conclusion
4875 \stex_if_smsmode:F {
4876 \seq_clear:N \l_tmpa_seq
4877 \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4878 \tl_if_empty:nF{ ##1 }{
4879 \stex_get_symbol:n { ##1 }
4880 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4881 \l_stex_get_symbol_uri_str
4882 }
4883 }
4884 }
4885 \exp_args:Nnnx
4886 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4887 \str_if_empty:NF \sexampletype {
4888 \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
4889 }

```

```

4890 \clist_set:No \l_tmpa_clist \sexamplotype
4891 \tl_clear:N \l_tmpa_tl
4892 \clist_map_inline:Nn \l_tmpa_clist {
4893   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4894     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4895   }
4896 }
4897 \tl_if_empty:NTF \l_tmpa_tl {
4898   \__stex_statements_sexample_start:
4899 }{
4900   \l_tmpa_tl
4901 }
4902 }
4903 \str_if_empty:NF \sexampleid {
4904   \stex_ref_new_doc_target:n \sexampleid
4905 }
4906 \stex_smsmode_do:
4907 ){
4908   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4909   \stex_if_smsmode:F {
4910     \clist_set:No \l_tmpa_clist \sexamplotype
4911     \tl_clear:N \l_tmpa_tl
4912     \clist_map_inline:Nn \l_tmpa_clist {
4913       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4914         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4915       }
4916     }
4917     \tl_if_empty:NTF \l_tmpa_tl {
4918       \__stex_statements_sexample_end:
4919     }{
4920       \l_tmpa_tl
4921     }
4922     \end{stex_annotate_env}
4923   }
4924 }

```

\stexpatchexample

```

4925
4926 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4927   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4928     (\sexamplename)
4929   }~}
4930 }
4931 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4932
4933 \newcommand\stexpatchexample[3]{} {
4934   \str_set:Nx \l_tmpa_str{ #1 }
4935   \str_if_empty:NTF \l_tmpa_str {
4936     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4937     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4938   }{
4939     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4940     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4941   }

```

4942 }

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

`\inlineex` inline:

```

4943 \keys_define:nn {stex / inlineex }{
4944   type      .str_set_x:N = \sexamplotype,
4945   id        .str_set_x:N = \sexampleid,
4946   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4947   name      .str_set_x:N = \sexamplename
4948 }
4949 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4950   \str_clear:N \sexamplotype
4951   \str_clear:N \sexampleid
4952   \str_clear:N \sexamplename
4953   \clist_clear:N \l__stex_statements_sexample_for_clist
4954   \keys_set:nn { stex / inlineex }{ #1 }
4955 }
4956 \NewDocumentCommand \inlineex { 0{ } m } {
4957   \begingroup
4958   \stex_reactivate_macro:N \premise
4959   \stex_reactivate_macro:N \conclusion
4960   \__stex_statements_inlineex_args:n{ #1 }
4961   \str_if_empty:NF \sexampleid {
4962     \stex_ref_new_doc_target:n \sexampleid
4963   }
4964   \stex_if_smsmode:TF{
4965     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4966   }{
4967     \seq_clear:N \l_tmpa_seq
4968     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4969       \tl_if_empty:nF{ ##1 }{
4970         \stex_get_symbol:n { ##1 }
4971         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4972           \l_stex_get_symbol_uri_str
4973         }
4974       }
4975     }
4976     \exp_args:Nnx
4977     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4978       \str_if_empty:NF \sexamplotype {
4979         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4980       }
4981       #2
4982       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4983     }
4984   }
4985   \endgroup
4986   \stex_smsmode_do:
4987 }

```

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

33.4 Logical Paragraphs

sparagraph

```

4988 \keys_define:nn { stex / sparagraph } {
4989   id       .str_set:x:N = \sparagraphid ,
4990   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4991   type     .str_set:x:N = \sparagraphtype ,
4992   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4993   from     .tl_set:N    = \sparagraphfrom ,
4994   to       .tl_set:N    = \sparagraphto ,
4995   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4996   name     .str_set:N   = \sparagraphname
4997 }
4998
4999 \cs_new_protected:Nn \stex_sparagraph_args:n {
5000   \tl_clear:N \l_stex_sparagraph_title_tl
5001   \tl_clear:N \sparagraphfrom
5002   \tl_clear:N \sparagraphto
5003   \tl_clear:N \l_stex_sparagraph_start_tl
5004   \str_clear:N \sparagraphid
5005   \str_clear:N \sparagraphtype
5006   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5007   \str_clear:N \sparagraphname
5008   \keys_set:nn { stex / sparagraph }{ #1 }
5009 }
5010 \newif\if@in@omtext\@in@omtextfalse
5011
5012 \NewDocumentEnvironment {sparagraph} { 0{} } {
5013   \stex_sparagraph_args:n { #1 }
5014   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5015     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5016   }{
5017     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5018   }
5019   \@in@omtexttrue
5020   \stex_if_smsmode:F {
5021     \seq_clear:N \l_tmpa_seq
5022     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5023       \tl_if_empty:NF{ ##1 }{
5024         \stex_get_symbol:n { ##1 }
5025         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5026           \l_stex_get_symbol_uri_str
5027         }
5028       }
5029     }
5030     \exp_args:Nnnx
5031     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5032     \str_if_empty:NF \sparagraphtype {
5033       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
5034     }
5035     \str_if_empty:NF \sparagraphfrom {
5036       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5037     }
5038     \str_if_empty:NF \sparagraphto {

```

```

5039     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5040 }
5041 \clist_set:No \l_tmpa_clist \sparagraphtype
5042 \tl_clear:N \l_tmpa_tl
5043 \clist_map_inline:Nn \sparagraphtype {
5044     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5045         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5046     }
5047 }
5048 \tl_if_empty:NTF \l_tmpa_tl {
5049     \__stex_statements_sparagraph_start:
5050 }{
5051     \l_tmpa_tl
5052 }
5053 }
5054 \clist_set:No \l_tmpa_clist \sparagraphtype
5055 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5056     {
5057         \stex_reactivate_macro:N \definiendum
5058         \stex_reactivate_macro:N \definame
5059         \stex_reactivate_macro:N \Definame
5060         \stex_reactivate_macro:N \premise
5061         \stex_reactivate_macro:N \definiens
5062     }
5063     \str_if_empty:NTF \sparagraphid {
5064         \str_if_empty:NTF \sparagraphname {
5065             \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5066                 \stex_ref_new_doc_target:n {}
5067             }
5068         } {
5069             \stex_ref_new_doc_target:n {}
5070         }
5071     } {
5072         \stex_ref_new_doc_target:n \sparagraphid
5073     }
5074     \exp_args:NNx
5075     \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5076         \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5077             \tl_if_empty:nF{ ##1 }{
5078                 \stex_get_symbol:n { ##1 }
5079                 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5080             }
5081         }
5082     }
5083     \stex_smsmode_do:
5084     \ignorespacesandpars
5085 }{
5086     \str_if_empty:NF \sparagraphname {
5087         \stex_symdecl_do:nn{}{\sparagraphname}
5088         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5089     }
5090     \stex_if_smsmode:F {
5091         \clist_set:No \l_tmpa_clist \sparagraphtype
5092         \tl_clear:N \l_tmpa_tl

```

```

5093 \clist_map_inline:Nn \l_tmpa_clist {
5094   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5095     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5096   }
5097 }
5098 \tl_if_empty:NTF \l_tmpa_tl {
5099   \__stex_statements_sparagraph_end:
5100 }{
5101   \l_tmpa_tl
5102 }
5103 \end{stex_annotate_env}
5104 }
5105 }

```

\stexpatchparagraph

```

5106
5107 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5108   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5109     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5110       \titleemph{\l_stex_sparagraph_title_tl}:~
5111     }
5112   }{
5113     \titleemph{\l_stex_sparagraph_start_tl}~
5114   }
5115 }
5116 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5117
5118 \newcommand\stexpatchparagraph[3]{} {
5119   \str_set:Nx \l_tmpa_str{ #1 }
5120   \str_if_empty:NTF \l_tmpa_str {
5121     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5122     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5123   }{
5124     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5125     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5126   }
5127 }
5128
5129 \keys_define:nn { stex / inlinepara } {
5130   id      .str_set_x:N = \sparagraphid ,
5131   type    .str_set_x:N = \sparagraphtype ,
5132   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5133   from    .tl_set:N    = \sparagraphfrom ,
5134   to      .tl_set:N    = \sparagraphto ,
5135   name    .str_set:N    = \sparagraphname
5136 }
5137 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5138   \tl_clear:N \sparagraphfrom
5139   \tl_clear:N \sparagraphto
5140   \str_clear:N \sparagraphid
5141   \str_clear:N \sparagraphtype
5142   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5143   \str_clear:N \sparagraphname
5144   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

5145 }
5146 \NewDocumentCommand \inlinepara { 0{} m } {
5147   \begingroup
5148   \_stex_statements_inlinepara_args:n{ #1 }
5149   \clist_set:No \l_tmpa_clist \sparagraphtype
5150   \str_if_empty:NTF \sparaagraphid {
5151     \str_if_empty:NTF \sparaagraphname {
5152       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5153         \stex_ref_new_doc_target:n {}
5154       }
5155     } {
5156       \stex_ref_new_doc_target:n {}
5157     }
5158   } {
5159     \stex_ref_new_doc_target:n \sparaagraphid
5160   }
5161   \stex_if_smsmode:TF{
5162     \str_if_empty:NF \sparaagraphname {
5163       \stex_symdecl_do:nn{}{\sparaagraphname}
5164       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5165     }
5166   }{
5167     \seq_clear:N \l_tmpa_seq
5168     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
5169       \tl_if_empty:nF{ ##1 }{
5170         \stex_get_symbol:n { ##1 }
5171         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5172           \l_stex_get_symbol_uri_str
5173         }
5174       }
5175     }
5176     \exp_args:Nnx
5177     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5178       \str_if_empty:NF \sparaagraphtype {
5179         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
5180       }
5181       \str_if_empty:NF \sparaagraphfrom {
5182         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5183       }
5184       \str_if_empty:NF \sparaagraphto {
5185         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5186       }
5187       \str_if_empty:NF \sparaagraphname {
5188         \stex_symdecl_do:nn{}{\sparaagraphname}
5189         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5190       }
5191       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5192         \clist_map_inline:Nn \l_tmpa_seq {
5193           \stex_ref_new_sym_target:n {##1}
5194         }
5195       }
5196     }
5197   }
5198 }

```



```

5199 \endgroup
5200 \stex_smsmode_do:
5201 }
5202
(End definition for \stexpatchparagraph. This function is documented on page ??.)
5203 </package>

```

Chapter 34

The Implementation

34.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).¹³

```
5204 <*package>
5205 <@@=stex_sproof>
5206
5207 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5208
```

34.2 Proofs

We first define some keys for the proof environment.

```
5209 \keys_define:nn { stex / spf } {
5210   id          .str_set_x:N = \spfid,
5211   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5212   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5213   proofend     .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5214   type        .str_set_x:N = \spftype,
5215   title       .tl_set:N    = \spftitle,
5216   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5217   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5218   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5219 }
5220 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5221   \str_clear:N \spfid
5222   \tl_clear:N \l__stex_sproof_spf_for_tl
5223   \tl_clear:N \l__stex_sproof_spf_from_tl
5224   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5225   \str_clear:N \spftype
5226   \tl_clear:N \spftitle
5227   \tl_clear:N \l__stex_sproof_spf_continues_tl
5228   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

5229 \tl_clear:N \l__stex_sproof_spf_method_tl
5230 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5231 \keys_set:nn { stex / spf }{ #1 }
5232 }

```

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

```

5233 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

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.

```

5234 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5235 \cs_new_protected:Npn \sproofnumber {
5236   \int_set:Nn \l_tmpa_int {1}
5237   \bool_while_do:nn {
5238     \int_compare_p:nNn {
5239       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5240     } > 0
5241   }{
5242     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5243     \int_incr:N \l_tmpa_int
5244   }
5245 }
5246 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5247   \int_set:Nn \l_tmpa_int {1}
5248   \bool_while_do:nn {
5249     \int_compare_p:nNn {
5250       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5251     } > 0
5252   }{
5253     \int_incr:N \l_tmpa_int
5254   }
5255   \int_compare:nNnF \l_tmpa_int = 1 {
5256     \int_decr:N \l_tmpa_int
5257   }
5258   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5259     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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

```

5260 }
5261 }
5262
5263 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5264   \int_set:Nn \l_tmpa_int {1}
5265   \bool_while_do:nn {
5266     \int_compare_p:nNn {
5267       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5268     } > 0
5269   }{
5270     \int_incr:N \l_tmpa_int
5271   }
5272   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5273 }
5274
5275 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5276   \int_set:Nn \l_tmpa_int {1}
5277   \bool_while_do:nn {
5278     \int_compare_p:nNn {
5279       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5280     } > 0
5281   }{
5282     \int_incr:N \l_tmpa_int
5283   }
5284   \int_decr:N \l_tmpa_int
5285   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5286 }

```

\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

```

5287 \def\sproof@box{
5288   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5289 }
5290 \def\sproofend{
5291   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5292     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5293   }
5294 }

```

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

spf@*@kw

```

5295 \def\spf@proofsketch@kw{Proof-Sketch}
5296 \def\spf@proof@kw{Proof}
5297 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5298 \AddToHook{begindocument}{
5299   \ltx@ifpackageloaded{babel}{
5300     \makeatletter
5301     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5302     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5303       \input{sproof-ngerman.ldf}

```

```

5304     }
5305     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5306       \input{sproof-finnish.ldf}
5307     }
5308     \clist_if_in:NnT \l_tmpa_clist {french}{
5309       \input{sproof-french.ldf}
5310     }
5311     \clist_if_in:NnT \l_tmpa_clist {russian}{
5312       \input{sproof-russian.ldf}
5313     }
5314     \makeatother
5315   }{}
5316 }

```

spfsketch

```

5317 \newcommand\spfsketch[2] [] {
5318   \beginingroup
5319   \let \premise \stex_proof_premise:
5320   \__stex_sproof_spf_args:n{#1}
5321   \stex_if_smsmode:TF {
5322     \str_if_empty:NF \spfid {
5323       \stex_ref_new_doc_target:n \spfid
5324     }
5325   }{
5326     \seq_clear:N \l_tmpa_seq
5327     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5328       \tl_if_empty:nF{ ##1 }{
5329         \stex_get_symbol:n { ##1 }
5330         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5331           \l_stex_get_symbol_uri_str
5332         }
5333       }
5334     }
5335     \exp_args:Nnx
5336     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5337       \str_if_empty:NF \spftype {
5338         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5339       }
5340       \clist_set:No \l_tmpa_clist \spftype
5341       \tl_set:Nn \l_tmpa_tl {
5342         \titleemph{
5343           \tl_if_empty:NTF \spftitle {
5344             \spf@proofsketch@kw
5345           }{
5346             \spftitle
5347           }
5348         }::~
5349       }
5350       \clist_map_inline:Nn \l_tmpa_clist {
5351         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5352           \tl_clear:N \l_tmpa_tl
5353         }
5354       }
5355       \str_if_empty:NF \spfid {

```

```

5356         \stex_ref_new_doc_target:n \spfid
5357     }
5358     \l_tmpa_tl #2 \sproofend
5359 }
5360 }
5361 \endgroup
5362 \stex_smsmode_do:
5363 }
5364

```

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

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

```

5365 \newenvironment{spfeq}[2][]{
5366     \__stex_sproof_spf_args:n{#1}
5367     \let \premise \stex_proof_premise:
5368     \stex_if_smsmode:TF {
5369         \str_if_empty:NF \spfid {
5370             \stex_ref_new_doc_target:n \spfid
5371         }
5372     }{
5373         \seq_clear:N \l_tmpa_seq
5374         \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5375             \tl_if_empty:NF{ ##1 }{
5376                 \stex_get_symbol:n { ##1 }
5377                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5378                     \l_stex_get_symbol_uri_str
5379                 }
5380             }
5381         }
5382         \exp_args:Nnnx
5383         \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5384         \str_if_empty:NF \spftype {
5385             \stex_annotate_invisible:nnn{type}{\spftype}{}
5386         }
5387
5388         \clist_set:No \l_tmpa_clist \spftype
5389         \tl_clear:N \l_tmpa_tl
5390         \clist_map_inline:Nn \l_tmpa_clist {
5391             \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5392                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5393             }
5394             \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5395                 \tl_set:Nn \l_tmpa_tl {\use:n{}}
5396             }
5397         }
5398         \tl_if_empty:NTF \l_tmpa_tl {
5399             \__stex_sproof_spfeq_start:
5400         }{
5401             \l_tmpa_tl
5402         }{-#2}

```

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

¹⁵EDNOTE: document above

```

5403 \str_if_empty:NF \spfid {
5404 \stex_ref_new_doc_target:n \spfid
5405 }
5406 \begin{displaymath}\begin{array}{rc1l}
5407 }
5408 \stex_smsmode_do:
5409 }{
5410 \stex_if_smsmode:F {
5411 \end{array}\end{displaymath}
5412 \clist_set:No \l_tmpa_clist \spftype
5413 \tl_clear:N \l_tmpa_tl
5414 \clist_map_inline:Nn \l_tmpa_clist {
5415 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5416 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5417 }
5418 }
5419 \tl_if_empty:NTF \l_tmpa_tl {
5420 \__stex_sproof_spfeq_end:
5421 }{
5422 \l_tmpa_tl
5423 }
5424 \end{stex_annotate_env}
5425 }
5426 }
5427
5428 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5429 \titleemph{
5430 \tl_if_empty:NTF \spftitle {
5431 \spf@proof@kw
5432 }{
5433 \spftitle
5434 }
5435 }:
5436 }
5437 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5438
5439 \newcommand\stexpatchspfeq[3] [] {
5440 \str_set:Nx \l_tmpa_str{ #1 }
5441 \str_if_empty:NTF \l_tmpa_str {
5442 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5443 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5444 }{
5445 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5446 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5447 }
5448 }
5449

```

(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.

```

5450 \newenvironment{sproof}[2] []{

```

```

5451 \let \premise \stex_proof_premise:
5452 \intarray_gzero:N \l__stex_sproof_counter_intarray
5453 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5454 \__stex_sproof_spf_args:n{#1}
5455 \stex_if_smsmode:TF {
5456   \str_if_empty:NF \spfid {
5457     \stex_ref_new_doc_target:n \spfid
5458   }
5459 }{
5460   \seq_clear:N \l_tmpa_seq
5461   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5462     \tl_if_empty:NF{ ##1 }{
5463       \stex_get_symbol:n { ##1 }
5464       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5465         \l_stex_get_symbol_uri_str
5466       }
5467     }
5468   }
5469   \exp_args:Nnnx
5470   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5471   \str_if_empty:NF \spftype {
5472     \stex_annotate_invisible:nnn{type}{\spftype}{}
5473   }
5474
5475   \clist_set:No \l_tmpa_clist \spftype
5476   \tl_clear:N \l_tmpa_tl
5477   \clist_map_inline:Nn \l_tmpa_clist {
5478     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5479       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5480     }
5481     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5482       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5483     }
5484   }
5485   \tl_if_empty:NTF \l_tmpa_tl {
5486     \__stex_sproof_sproof_start:
5487   }{
5488     \l_tmpa_tl
5489   }{~#2}
5490   \str_if_empty:NF \spfid {
5491     \stex_ref_new_doc_target:n \spfid
5492   }
5493   \begin{description}
5494 }
5495 \stex_smsmode_do:
5496 }{
5497   \stex_if_smsmode:F{
5498     \end{description}
5499     \clist_set:No \l_tmpa_clist \spftype
5500     \tl_clear:N \l_tmpa_tl
5501     \clist_map_inline:Nn \l_tmpa_clist {
5502       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5503         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5504       }

```



```

5505     }
5506     \tl_if_empty:NTF \l_tmpa_tl {
5507       \__stex_sproof_sproof_end:
5508     }{
5509       \l_tmpa_tl
5510     }
5511     \end{stex_annotate_env}
5512   }
5513 }
5514
5515 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5516   \par\noindent\titleemph{
5517     \tl_if_empty:NTF \spftype {
5518       \spf@proof@kw
5519     }{
5520       \spftype
5521     }
5522   }:
5523 }
5524 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5525
5526 \newcommand\stexpatchsproof[3] [] {
5527   \str_set:Nx \l_tmpa_str{ #1 }
5528   \str_if_empty:NTF \l_tmpa_str {
5529     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5530     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5531   }{
5532     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5533     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5534   }
5535 }

```

\spfidea

```

5536 \newcommand\spfidea[2] []{
5537   \__stex_sproof_spf_args:n{#1}
5538   \titleemph{
5539     \tl_if_empty:NTF \spftype {Proof~Idea}{
5540       \spftype
5541     }:
5542   }~#2
5543   \sproofend
5544 }

```

(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

```

5545 \newenvironment{spfstep}[1] []{
5546   \__stex_sproof_spf_args:n{#1}
5547   \stex_if_smsmode:TF {

```

```

5548 \str_if_empty:NF \spfid {
5549 \stex_ref_new_doc_target:n \spfid
5550 }
5551 }{
5552 \@in@omtexttrue
5553 \seq_clear:N \l_tmpa_seq
5554 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5555 \tl_if_empty:nF{ ##1 }{
5556 \stex_get_symbol:n { ##1 }
5557 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5558 \l_stex_get_symbol_uri_str
5559 }
5560 }
5561 }
5562 \exp_args:Nnnx
5563 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5564 \str_if_empty:NF \spftype {
5565 \stex_annotate_invisible:nnn{type}{\spftype}{}
5566 }
5567 \clist_set:No \l_tmpa_clist \spftype
5568 \tl_set:Nn \l_tmpa_tl {
5569 \item[\sproofnumber]
5570 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5571 }
5572 \clist_map_inline:Nn \l_tmpa_clist {
5573 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5574 \tl_clear:N \l_tmpa_tl
5575 }
5576 }
5577 \l_tmpa_tl
5578 \tl_if_empty:NF \spftitle {
5579 {(\titleemph{\spftitle})\enspace}
5580 }
5581 \str_if_empty:NF \spfid {
5582 \stex_ref_new_doc_target:n \spfid
5583 }
5584 }
5585 \stex_smsmode_do:
5586 \ignorespacesandpars
5587 }{
5588 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5589 \__stex_sproof_inc_counter:
5590 }
5591 \stex_if_smsmode:F {
5592 \end{stex_annotate_env}
5593 }
5594 }

```

sproofcomment

```

5595 \newenvironment{sproofcomment}[1][]{
5596 \__stex_sproof_spf_args:n{#1}
5597 \clist_set:No \l_tmpa_clist \spftype
5598 \tl_set:Nn \l_tmpa_tl {
5599 \item[\sproofnumber]

```

```

5600 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5601 }
5602 \clist_map_inline:Nn \l_tmpa_clist {
5603   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5604     \tl_clear:N \l_tmpa_tl
5605   }
5606 }
5607 \l_tmpa_tl
5608 }{
5609   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5610     \__stex_sproof_inc_counter:
5611   }
5612 }

```

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) `proproof` environment is started.

```

5613 \newenvironment{subproof}[2][]{
5614   \__stex_sproof_spf_args:n{#1}
5615   \stex_if_smsmode:TF{
5616     \str_if_empty:NF \spfid {
5617       \stex_ref_new_doc_target:n \spfid
5618     }
5619   }{
5620     \seq_clear:N \l_tmpa_seq
5621     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5622       \tl_if_empty:nF{ ##1 }{
5623         \stex_get_symbol:n { ##1 }
5624         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5625           \l_stex_get_symbol_uri_str
5626         }
5627       }
5628     }
5629     \exp_args:Nnnx
5630     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5631     \str_if_empty:NF \spftype {
5632       \stex_annotate_invisible:nnn{type}{\spftype}{\}
5633     }
5634
5635     \clist_set:No \l_tmpa_clist \spftype
5636     \tl_set:Nn \l_tmpa_tl {
5637       \item[\sproofnumber]
5638       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5639     }
5640     \clist_map_inline:Nn \l_tmpa_clist {
5641       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5642         \tl_clear:N \l_tmpa_tl
5643       }
5644     }
5645     \l_tmpa_tl
5646     \tl_if_empty:NF \spftitle {
5647       {(\titleemph{\spftitle})\enspace}
5648     }

```

```

5649     {~#2}
5650     \str_if_empty:NF \spfid {
5651       \stex_ref_new_doc_target:n \spfid
5652     }
5653   }
5654   \__stex_sproof_add_counter:
5655   \stex_smsmode_do:
5656 }{
5657   \__stex_sproof_remove_counter:
5658   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5659     \__stex_sproof_inc_counter:
5660   }
5661   \stex_if_smsmode:F{
5662     \end{stex_annotate_env}
5663   }
5664 }

```

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

```

5665 \newenvironment{spfcases}[2][]{
5666   \tl_if_empty:nTF{#1}{
5667     \begin{subproof}[method=by-cases]{#2}
5668   }{
5669     \begin{subproof}[#1,method=by-cases]{#2}
5670   }
5671 }{
5672   \end{subproof}
5673 }

```

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

```

5674 \newenvironment{spfcase}[2][]{
5675   \__stex_sproof_spf_args:n{#1}
5676   \stex_if_smsmode:TF {
5677     \str_if_empty:NF \spfid {
5678       \stex_ref_new_doc_target:n \spfid
5679     }
5680   }{
5681     \seq_clear:N \l_tmpa_seq
5682     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5683       \tl_if_empty:nF{ ##1 }{
5684         \stex_get_symbol:n { ##1 }
5685         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5686           \l_stex_get_symbol_uri_str
5687         }
5688       }
5689     }
5690     \exp_args:Nnnx
5691     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5692     \str_if_empty:NF \spftype {
5693       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5694   }
5695   \clist_set:Nn \l_tmpa_clist \spftype
5696   \tl_set:Nn \l_tmpa_tl {
5697     \item[\sproofnumber]

```

```

5698     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5699   }
5700   \clist_map_inline:Nn \l_tmpa_clist {
5701     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5702       \tl_clear:N \l_tmpa_tl
5703     }
5704   }
5705   \l_tmpa_tl
5706   \tl_if_empty:nF{#2}{
5707     \titleemph{#2}:~
5708   }
5709 }
5710 \__stex_sproof_add_counter:
5711 \stex_smsmode_do:
5712 ){
5713   \__stex_sproof_remove_counter:
5714   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5715     \__stex_sproof_inc_counter:
5716   }
5717   \stex_if_smsmode:F{
5718     \clist_set:No \l_tmpa_clist \spftype
5719     \tl_set:Nn \l_tmpa_tl{\sproofend}
5720     \clist_map_inline:Nn \l_tmpa_clist {
5721       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5722         \tl_clear:N \l_tmpa_tl
5723       }
5724     }
5725     \l_tmpa_tl
5726     \end{stex_annotate_env}
5727   }
5728 }

```

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

```

5729 \newcommand\spfcasesketch[3][]{
5730   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5731 }

```

34.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

5732 \keys_define:nn { stex / just }{
5733   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5734   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
5735   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
5736   args     .tl_set:N   = \l__stex_sproof_just_args_tl
5737 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.¹⁶

¹⁶EdNOTE: need to do something about the premise in draft mode.

justification

```
5738 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5739 \newcommand\stex_proof_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.

```
5740 \newcommand\justarg[2] [] {#2}
```

```
5741 \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 35

STEX -Others Implementation

```
5742 <*package>
5743
5744 %%%%%%%%%% others.dtx %%%%%%%%%%
5745
5746 <@@=stex_others>
    Warnings and error messages
5747 % None

\MSC Math subject classifier

5748 \NewDocumentCommand \MSC {m} {
5749 % TODO
5750 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5751 \@ifpackageloaded{tikzinput}{
5752 \RequirePackage{stex-tikzinput}
5753 }{}
5754 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
5755 <*package>
5756 <@@=stex_modules>
5757
5758 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5759
5760 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5761 \begingroup
5762 \stex_module_setup:nn{
5763   ns=\c_stex_metatheory_ns_str,
5764   meta=NONE
5765 }{Metatheory}
5766 \stex_reactivate_macro:N \symdecl
5767 \stex_reactivate_macro:N \notation
5768 \stex_reactivate_macro:N \symdef
5769 \ExplSyntaxOff
5770 \csname stex_suppress_html:n\endcsname{
5771   % is-a (a:A, a \in A, a is an A, etc.)
5772   \symdecl{isa}[args=ai]
5773   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5774   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5775   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5776
5777   % bind (\forall, \Pi, \lambda etc.)
5778   \symdecl{bind}[args=Bi]
5779   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5780   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5781   \notation{bind}[deffun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5782
5783   % implicit bind
5784   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5785
5786   % dummy variable
5787   \symdecl{dummyvar}
5788   \notation{dummyvar}[underscore]{\comp\_}
5789   \notation{dummyvar}[dot]{\comp\cdot}
```



```

5790 \notation{dummyvar}[dash]{\comp{\rm --}}
5791
5792 %fromto (function space, Hom-set, implication etc.)
5793 \symdecl{fromto}[args=ai]
5794 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5795 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5796
5797 % mapto (lambda etc.)
5798 \symdecl{mapto}[args=Bi]
5799 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5800 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5801 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5802
5803 % function/operator application
5804 \symdecl{apply}[args=ia]
5805 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5806 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5807
5808 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5809 \symdecl{metacollection}
5810 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5811 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5812
5813 % collection of propositions/booleans/truth values
5814 \symdecl{prop}[name=proposition]
5815 \notation{prop}[prop]{\comp{\rm prop}}
5816 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5817
5818 % sequences
5819 \symdecl{seqtype}[args=1]
5820 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5821
5822 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{1}_{#2}}
5823 \notation{sequence-index}[ui,prec=nobrackets]{#{1}^{\comp\ast}}
5824
5825 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5826 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5827 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5828
5829 % letin (‘‘let’’, local definitions, variable substitution)
5830 \symdecl{letin}[args=bii]
5831 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
5832 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5833 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5834
5835 % structures
5836 \symdecl*{module-type}[args=1]
5837 \notation{module-type}{\mathtt{MOD} #1}
5838 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5839 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5840
5841 }
5842 \ExplSyntaxOn
5843 \stex_add_to_current_module:n{

```

```

5844 \let\nappa\apply
5845 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5846 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5847 \def\livar{\csname sequence-index\endcsname[li]}
5848 \def\uivar{\csname sequence-index\endcsname[ui]}
5849 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5850 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5851 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5852 }
5853 \__stex_modules_end_module:
5854 \endgroup
5855 </package>

```

Chapter 37

Tikzinput Implementation

```
5856 <*package>
5857
5858 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5859
5860 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5861 \RequirePackage{l3keys2e}
5862
5863 \keys_define:nn { tikzinput } {
5864   image .bool_set:N = \c_tikzinput_image_bool,
5865   image .default:n = false ,
5866   unknown .code:n = {}
5867 }
5868
5869 \ProcessKeysOptions { tikzinput }
5870
5871 \bool_if:NTF \c_tikzinput_image_bool {
5872   \RequirePackage{graphicx}
5873
5874   \providecommand\usetikzlibrary[]{}
5875   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
5876 }{
5877   \RequirePackage{tikz}
5878   \RequirePackage{standalone}
5879
5880   \newcommand \tikzinput [2] [] {
5881     \setkeys{Gin}{#1}
5882     \ifx \Gin@ewidth \Gin@exclamation
5883       \ifx \Gin@eheight \Gin@exclamation
5884         \input { #2 }
5885       \else
5886         \resizebox{!}{ \Gin@eheight }{
5887           \input { #2 }
5888         }
5889       \fi
5890     \else
5891       \ifx \Gin@eheight \Gin@exclamation
5892         \resizebox{ \Gin@ewidth }{!}{
5893           \input { #2 }
```

```

5894     }
5895     \else
5896         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5897             \input { #2 }
5898         }
5899     \fi
5900 \fi
5901 }
5902 }
5903
5904 \newcommand \ctikzinput [2] [] {
5905     \begin{center}
5906         \tikzinput [ #1 ] { #2 }
5907     \end{center}
5908 }
5909
5910 \@ifpackageloaded{stex}{
5911     \RequirePackage{stex-tikzinput}
5912 }{}
5913
5914 </package>
5915 <*stex>
5916 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5917 \RequirePackage{stex}
5918 \RequirePackage{tikzinput}
5919
5920 \newcommand\mhtikzinput [2] [] {%
5921     \def\Gin@mhrepos{}\setkeys{Gin}{ #1 }%
5922     \stex_in_repository:nn\Gin@mhrepos{
5923         \tikzinput [ #1 ] {\mhp{ ##1 } { #2 }}
5924     }
5925 }
5926 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5927 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

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

```
5928 \*cls)
5929 \@@=document_structure)
5930 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5931 \RequirePackage{13keys2e}
```

38.2 Class Options

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

`\omdoc@cls@class`

```
5932 \keys_define:nn{ document-structure / pkg }{
5933   class      .str_set_x:N = \c_document_structure_class_str,
5934   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5935   report     .code:n      = {
5936     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5937     \str_set:Nn \c_document_structure_class_str {report}
5938   },
5939   book       .code:n      = {
5940     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5941     \str_set:Nn \c_document_structure_class_str {book}
5942   },
5943   bookpart   .code:n      = {
5944     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5945     \str_set:Nn \c_document_structure_class_str {book}
5946     \str_set:Nn \c_document_structure_topsect_str {chapter}
5947   },
```

```

5948 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5949 unknown     .code:n      = {
5950   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5951 }
5952 }
5953 \ProcessKeysOptions{ document-structure / pkg }
5954 \str_if_empty:NT \c_document_structure_class_str {
5955   \str_set:Nn \c_document_structure_class_str {article}
5956 }
5957 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5958   {\c_document_structure_class_str}
5959

```

38.3 Beefing up the document environment

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

```

5960 \RequirePackage{document-structure}
5961 \bool_if:NF \c_document_structure_minimal_bool {

```

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

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.¹⁷

```

5962 \keys_define:nn { document-structure / document }{
5963   id .str_set_x:N = \c_document_structure_document_id_str
5964 }
5965 \let\__document_structure_orig_document=\document
5966 \renewcommand{\document}[1][]{
5967   \keys_set:nn{ document-structure / document }{ #1 }
5968   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5969   \__document_structure_orig_document
5970 }

```

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

```

5971 }
5972 \</cls>

```

38.4 Implementation: document-structure Package

```

5973 <*package>
5974 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
5975 \RequirePackage{l3keys2e}

```

38.5 Package Options

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

¹⁷EDNOTE: faking documentkeys for now. @HANG, please implement

```

5976
5977 \keys_define:nn{ document-structure / pkg }{
5978   class      .str_set_x:N = \c_document_structure_class_str,
5979   topsect     .str_set_x:N = \c_document_structure_topsect_str,
5980   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5981 }
5982 \ProcessKeysOptions{ document-structure / pkg }
5983 \str_if_empty:NT \c_document_structure_class_str {
5984   \str_set:Nn \c_document_structure_class_str {article}
5985 }
5986 \str_if_empty:NT \c_document_structure_topsect_str {
5987   \str_set:Nn \c_document_structure_topsect_str {section}
5988 }

```

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

```

5989 \RequirePackage{xspace}
5990 \RequirePackage{comment}
5991 \AddToHook{begindocument}{
5992   \ltx@ifpackageloaded{babel}{
5993     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5994     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5995       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
5996     }
5997   }{}
5998 }

```

`\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.

```

5999 \int_new:N \l_document_structure_section_level_int
6000 \str_case:VnF \c_document_structure_topsect_str {
6001   {part}}{
6002     \int_set:Nn \l_document_structure_section_level_int {0}
6003   }
6004   {chapter}{
6005     \int_set:Nn \l_document_structure_section_level_int {1}
6006   }
6007 }{
6008   \str_case:VnF \c_document_structure_class_str {
6009     {book}}{
6010       \int_set:Nn \l_document_structure_section_level_int {0}
6011     }
6012     {report}}{
6013       \int_set:Nn \l_document_structure_section_level_int {0}
6014     }
6015   }{
6016     \int_set:Nn \l_document_structure_section_level_int {2}
6017   }
6018 }

```

38.6 Document Structure

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

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁸

EdN:18

```
6019 \def\current@section@level{document}%
6020 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6021 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6022 \cs_new_protected:Npn \skipomgroup {
6023   \ifcase\l_document_structure_section_level_int
6024   \or\stepcounter{part}
6025   \or\stepcounter{chapter}
6026   \or\stepcounter{section}
6027   \or\stepcounter{subsection}
6028   \or\stepcounter{subsubsection}
6029   \or\stepcounter{paragraph}
6030   \or\stepcounter{subparagraph}
6031   \fi
6032 }
```

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

`blindfragment`

```
6033 \newcommand\at@begin@blindomgroup[1]{%
6034   \newenvironment{blindfragment}
6035   {
6036     \int_incr:N\l_document_structure_section_level_int
6037     \at@begin@blindomgroup\l_document_structure_section_level_int
6038   }{}}
```

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

```
6039 \newcommand\omgroup@nonum[2]{
6040   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6041   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6042 }
```

(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.

```
6043 \newcommand\omgroup@num[2]{
```

¹⁸EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.


```

6044 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6045   \@nameuse{#1}{#2}
6046 }{
6047   \cs_if_exist:NTF\rdfmata@sectioning{
6048     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6049   }{
6050     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6051   }
6052 }
6053 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6054 }

```

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

sfragment

```

6055 \keys_define:nn { document-structure / omgroup }{
6056   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6057   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6058   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6059   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6060   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6061   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6062   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6063   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6064   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6065   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6066 }
6067 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
6068   \str_clear:N \l__document_structure_omgroup_id_str
6069   \str_clear:N \l__document_structure_omgroup_date_str
6070   \clist_clear:N \l__document_structure_omgroup_creators_clist
6071   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6072   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6073   \tl_clear:N \l__document_structure_omgroup_type_tl
6074   \tl_clear:N \l__document_structure_omgroup_short_tl
6075   \tl_clear:N \l__document_structure_omgroup_display_tl
6076   \tl_clear:N \l__document_structure_omgroup_intro_tl
6077   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6078   \keys_set:nn { document-structure / omgroup } { #1 }
6079 }

```

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.

```

6080 \newif\if@mainmatter\@mainmattertrue
6081 \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.

```

6082 \keys_define:nn { document-structure / sectioning }{
6083   name .str_set_x:N = \l__document_structure_sect_name_str ,
6084   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6085   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6086   clear .default:n = {true} ,
6087   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6088   num      .default:n      = {true}
6089 }
6090 \cs_new_protected:Nn \__document_structure_sect_args:n {
6091   \str_clear:N \l__document_structure_sect_name_str
6092   \str_clear:N \l__document_structure_sect_ref_str
6093   \bool_set_false:N \l__document_structure_sect_clear_bool
6094   \bool_set_false:N \l__document_structure_sect_num_bool
6095   \keys_set:nn { document-structure / sectioning } { #1 }
6096 }
6097 \newcommand\omdoc@sectioning[3][]{
6098   \__document_structure_sect_args:n {#1}
6099   \let\omdoc@sect@name\l__document_structure_sect_name_str
6100   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6101   \if@mainmatter% numbering not overridden by frontmatter, etc.
6102     \bool_if:NTF \l__document_structure_sect_num_bool {
6103       \omgroup@num{#2}{#3}
6104     }{
6105       \omgroup@nonum{#2}{#3}
6106     }
6107     \def\current@section@level{\omdoc@sect@name}
6108   \else
6109     \omgroup@nonum{#2}{#3}
6110   \fi
6111 }% 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.

```

6112 \newcommand\omgroup@redefine@addtocontents[1]{%
6113   %\edef\__document_structureimport{#1}%
6114   %\@for\@I:=\__document_structureimport\do{%
6115     %\edef\@path{\csname module@\@I @path\endcsname}%
6116     %\@ifundefined{tf@toc}\relax%
6117     %{\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6118   %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6119   %\def\addcontentsline##1##2##3{%
6120     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6121   %\else% hyperref.sty not loaded
6122   %\def\addcontentsline##1##2##3{%
6123     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6124   %\fi
6125 }% hypreref.sty loaded?

```

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

```

6126 \newenvironment{sfragment}[2][]{% keys, title
6127 {
6128   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

6129   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6130     \omgroup@redefine@addtocontents{
6131       %\@ifundefined{module@id}\used@modules%

```

```

6132     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6133   }
6134 }

```

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

```

6135 \int_incr:N\l_document_structure_section_level_int
6136 \ifcase\l_document_structure_section_level_int
6137   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6138   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6139   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6140   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6141   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6142   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6143   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6144 \fi
6145 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6146 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6147   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6148 }
6149 }% for customization
6150 {}

```

and finally, we localize the sections

```

6151 \newcommand\omdoc@part@kw{Part}
6152 \newcommand\omdoc@chapter@kw{Chapter}
6153 \newcommand\omdoc@section@kw{Section}
6154 \newcommand\omdoc@subsection@kw{Subsection}
6155 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6156 \newcommand\omdoc@paragraph@kw{paragraph}
6157 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

38.7 Front and Backmatter

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

`\printindex`

```

6158 \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).

```

6159 \cs_if_exist:NTF\frontmatter{
6160   \let\__document_structure_orig_frontmatter\frontmatter
6161   \let\frontmatter\relax
6162 }{
6163   \tl_set:Nn\__document_structure_orig_frontmatter{
6164     \clearpage
6165     \@mainmatterfalse
6166     \pagenumbering{roman}

```

```

6167 }
6168 }
6169 \cs_if_exist:NTF\backmatter{
6170   \let\__document_structure_orig_backmatter\backmatter
6171   \let\backmatter\relax
6172 }{
6173   \tl_set:Nn\__document_structure_orig_backmatter{
6174     \clearpage
6175     \@mainmatterfalse
6176     \pagenumbering{roman}
6177   }
6178 }

```

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.

```

6179 \newenvironment{frontmatter}{
6180   \__document_structure_orig_frontmatter
6181 }{
6182   \cs_if_exist:NTF\mainmatter{
6183     \mainmatter
6184   }{
6185     \clearpage
6186     \@mainmattertrue
6187     \pagenumbering{arabic}
6188   }
6189 }

```

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

```

6190 \newenvironment{backmatter}{
6191   \__document_structure_orig_backmatter
6192 }{
6193   \cs_if_exist:NTF\mainmatter{
6194     \mainmatter
6195   }{
6196     \clearpage
6197     \@mainmattertrue
6198     \pagenumbering{arabic}
6199   }
6200 }

```

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

```

6201 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6202 \def \c__document_structure_document_str{document}
6203 \newcommand\afterprematurestop{}
6204 \def\prematurestop@endomgroup{
6205   \unless\ifx\@currenvir\c__document_structure_document_str
6206     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6207     \expandafter\prematurestop@endomgroup

```

```

6208 \fi
6209 }
6210 \providecommand\prematurestop{
6211 \message{Stopping~sTeX~processing~prematurely}
6212 \prematurestop@endomgroup
6213 \afterprematurestop
6214 \end{document}
6215 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

6216 \RequirePackage{etoolbox}
6217 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6218 \newrobustcmd\useSGvar[1]{%
6219 \@ifundefined{sTeX@Gvar@#1}
6220 {\PackageError{document-structure}
6221 {The sTeX Global variable #1 is undefined}
6222 {set it with \protect\setSGvar}}
6223 \@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.

```

6224 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6225 \@ifundefined{sTeX@Gvar@#1}
6226 {\PackageError{document-structure}
6227 {The sTeX Global variable #1 is undefined}
6228 {set it with \protect\setSGvar}}
6229 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 39

NotesSlides – Implementation

39.1 Class and Package Options

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

```
6230 \*cls)
6231 \@@=notesslides)
6232 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6233 \RequirePackage{13keys2e}
6234
6235 \keys_define:nn{notesslides / cls}{
6236   class .code:n = {
6237     \PassOptionsToClass{\CurrentOption}{document-structure}
6238     \str_if_eq:nnT{#1}{book}{
6239       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6240     }
6241     \str_if_eq:nnT{#1}{report}{
6242       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6243     }
6244   },
6245   notes .bool_set:N = \c__notesslides_notes_bool ,
6246   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6247   unknown .code:n = {
6248     \PassOptionsToClass{\CurrentOption}{document-structure}
6249     \PassOptionsToClass{\CurrentOption}{beamer}
6250     \PassOptionsToPackage{\CurrentOption}{notesslides}
6251   }
6252 }
6253 \ProcessKeysOptions{ notesslides / cls }
6254 \bool_if:NTF \c__notesslides_notes_bool {
6255   \PassOptionsToPackage{notes=true}{notesslides}
6256 }{
6257   \PassOptionsToPackage{notes=false}{notesslides}
6258 }
6259 \</cls)
```

now we do the same for the notesslides package.

```

6260 <*package>
6261 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6262 \RequirePackage{13keys2e}
6263
6264 \keys_define:nn{notesslides / pkg}{
6265   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6266   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6267   notes        .bool_set:N = \c_notesslides_notes_bool ,
6268   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6269   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6270   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
6271   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
6272   nopproblems  .bool_set:N = \c_notesslides_nopproblems_bool,
6273   unknown      .code:n      = {
6274     \PassOptionsToClass{\CurrentOption}{stex}
6275     \PassOptionsToClass{\CurrentOption}{tikzinput}
6276   }
6277 }
6278 \ProcessKeysOptions{ notesslides / pkg }
6279 \newif\ifnotes
6280 \bool_if:NTF \c_notesslides_notes_bool {
6281   \notesttrue
6282 }{
6283   \notesfalse
6284 }
6285

```

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

```

6286 \str_if_empty:NTF \c_notesslides_topsect_str {
6287   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
6288 }{
6289   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
6290 }
6291 </package>

```

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

```

6292 <*cls>
6293 \bool_if:NTF \c_notesslides_notes_bool {
6294   \LoadClass{document-structure}
6295 }{
6296   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6297   \newcounter{Item}
6298   \newcounter{paragraph}
6299   \newcounter{subparagraph}
6300   \newcounter{Hfootnote}
6301   \RequirePackage{document-structure}
6302 }

```

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

```

6303 \RequirePackage{notesslides}
6304 </cls>

```

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

```

6305 \*package>
6306 \bool_if:NT \c__notesslides_notes_bool {
6307   \RequirePackage{a4wide}
6308   \RequirePackage{marginnote}
6309   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6310   \RequirePackage{mdframed}
6311   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6312   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6313 }
6314 \RequirePackage{stex-tikzinput}
6315 \RequirePackage{etoolbox}
6316 \RequirePackage{amssymb}
6317 \RequirePackage{amsmath}
6318 \RequirePackage{comment}
6319 \RequirePackage{textcomp}
6320 \RequirePackage{url}
6321 \RequirePackage{graphicx}
6322 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.¹⁹

```

6323 \bool_if:NT \c__notesslides_notes_bool {
6324   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6325 }
6326
6327
6328 \NewDocumentCommand \libusetheme {0{} m} {
6329   \bool_if:NTF \c__notesslides_notes_bool {
6330     \libusepackage[#1]{beamernotestheme#2}
6331   }{
6332     \libusepackage[#1]{beamertheme#2}
6333   }
6334 }

```

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

```

6335 \newcounter{slide}
6336 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6337 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

¹⁹EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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.

```

6338 \bool_if:NTF \c__notesslides_notes_bool {
6339   \renewenvironment{note}{\ignorespaces}{}
6340 }{
6341   \excludecomment{note}
6342 }

```

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.

```

6343 \bool_if:NT \c__notesslides_notes_bool {
6344   \newlength{\slideframewidth}
6345   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6346 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6347   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6348     \bool_set_true:N #1
6349   }{
6350     \bool_set_false:N #1
6351   }
6352 }
6353 \keys_define:nn{notesslides / frame}{
6354   label .str_set_x:N = \l__notesslides_frame_label_str,
6355   allowframebreaks .code:n = {
6356     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6357   },
6358   allowdisplaybreaks .code:n = {
6359     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6360   },
6361   fragile .code:n = {
6362     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6363   },
6364   shrink .code:n = {
6365     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6366   },
6367   squeeze .code:n = {
6368     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6369   },
6370   t .code:n = {
6371     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6372   },
6373 }
6374 \cs_new_protected:Nn \__notesslides_frame_args:n {
6375   \str_clear:N \l__notesslides_frame_label_str
6376   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6377   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6378   \bool_set_true:N \l__notesslides_frame_fragile_bool
6379   \bool_set_true:N \l__notesslides_frame_shrink_bool
6380   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6381   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6382     \keys_set:nn { notesslides / frame }{ #1 }
6383   }

```

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

```

6384   \renewenvironment{frame}[1][]{
6385     \__notesslides_frame_args:n{#1}
6386     \sffamily
6387     \stepcounter{slide}
6388     \def\@currentlabel{\theslide}
6389     \str_if_empty:NF \l__notesslides_frame_label_str {
6390       \label{\l__notesslides_frame_label_str}
6391     }

```

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

```

6392     \def\itemize@level{outer}
6393     \def\itemize@outer{outer}
6394     \def\itemize@inner{inner}
6395     \renewcommand\newpage{\addtocounter{framenumber}{1}}
6396     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6397     \renewenvironment{itemize}{
6398       \ifx\itemize@level\itemize@outer
6399         \def\itemize@label{$\rhd$}
6400       \fi
6401       \ifx\itemize@level\itemize@inner
6402         \def\itemize@label{$\scriptstyle\rhd$}
6403       \fi
6404       \begin{list}
6405         {\itemize@label}
6406         {\setlength{\labelsep}{.3em}
6407          \setlength{\labelwidth}{.5em}
6408          \setlength{\leftmargin}{1.5em}
6409         }
6410       \edef\itemize@level{\itemize@inner}
6411     }{
6412       \end{list}
6413     }

```

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

```

6414     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6415   }{
6416     \medskip\miko@slidelabel\end{mdframed}
6417   }

```

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

`\frametitle`

```

6418   \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6419 }

```

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

EdN:20

`\pause`

```

20
6420 \bool_if:NT \c__notesslides_notes_bool {
6421   \newcommand\pause{}
6422 }

```

²⁰EdNOTE: MK: fake it in notes mode for now

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

nparagraph

```
6423 \bool_if:NTF \c__notesslides_notes_bool {
6424   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6425 }{
6426   \excludecomment{nparagraph}
6427 }
```

nfragment

```
6428 \bool_if:NTF \c__notesslides_notes_bool {
6429   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6430 }{
6431   \excludecomment{nfragment}
6432 }
```

ndefinition

```
6433 \bool_if:NTF \c__notesslides_notes_bool {
6434   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6435 }{
6436   \excludecomment{ndefinition}
6437 }
```

nassertion

```
6438 \bool_if:NTF \c__notesslides_notes_bool {
6439   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6440 }{
6441   \excludecomment{nassertion}
6442 }
```

nsproof

```
6443 \bool_if:NTF \c__notesslides_notes_bool {
6444   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6445 }{
6446   \excludecomment{nsproof}
6447 }
```

nexample

```
6448 \bool_if:NTF \c__notesslides_notes_bool {
6449   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6450 }{
6451   \excludecomment{nexample}
6452 }
```

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

```
6453 \def\inputref@preskip{\smallskip}
6454 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6455 \let\orig@inputref\inputref
6456 \def\inputref{\@ifstar\ninputref\orig@inputref}
6457 \newcommand\ninputref[2][]{
6458   \bool_if:NT \c__notesslides_notes_bool {
6459     \orig@inputref[#1]{#2}
6460   }
6461 }
```

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

39.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the \TeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
6462 \newlength{\slidelogoheight}
6463
6464 \bool_if:NTF \c__notesslides_notes_bool {
6465   \setlength{\slidelogoheight}{.4cm}
6466 }{
6467   \setlength{\slidelogoheight}{1cm}
6468 }
6469 \newsavebox{\slidelogo}
6470 \sbox{\slidelogo}{\sTeX}
6471 \newrobustcmd{\setslidelogo}[1]{\def\source{#1}}
6472 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6473 }
```

(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.

```
6474 \def\source{Michael Kohlhase}% customize locally
6475 \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.

```
6476 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6477 \newsavebox{\cclogo}
6478 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6479 \newif\ifcchref\cchreffalse
6480 \AtBeginDocument{
6481   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6482 }
6483 \def\licensing{
6484   \ifcchref
```

```

6485     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6486   \else
6487     {\usebox{\cclogo}}
6488   \fi
6489 }
6490 \newrobustcmd{\setlicensing}[2][]{
6491   \def\@url{\#1}
6492   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
6493   \ifx\@url\@empty
6494     \def\licensing{\usebox{\cclogo}}
6495   \else
6496     \def\licensing{
6497       \ifcchref
6498         \href{\#1}{\usebox{\cclogo}}
6499       \else
6500         {\usebox{\cclogo}}
6501       \fi
6502     }
6503   \fi
6504 }

```

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

EdN:21

\slidelabel Now, we set up the slide label for the article mode.²¹

```

6505 \newrobustcmd\miko@slidelabel{
6506   \vbox to \slidelogoheight{
6507     \vss\hbox to \slidewidth
6508     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6509   }
6510 }

```

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

39.4 Frame Images

\frameimage We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

6511 \def\Gin@mhrepos{}
6512 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{\#1}}
6513 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
6514 \newrobustcmd\frameimage[2][]{
6515   \stepcounter{slide}
6516   \bool_if:NT \c__notesslides_frameimages_bool {
6517     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
6518     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6519     \begin{center}
6520       \bool_if:NTF \c__notesslides_fiboxed_bool {
6521         \fbox{
6522           \ifx\Gin@ewidth\@empty
6523             \ifx\Gin@mhrepos\@empty
6524               \mhgraphics[width=\slidewidth,\#1]{\#2}
6525             \else

```

²¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6526         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6527     \fi
6528     \else% Gin@ewidth empty
6529         \ifx\Gin@mhrepos\@empty
6530             \mhgraphics[#1]{#2}
6531         \else
6532             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6533         \fi
6534     \fi% Gin@ewidth empty
6535 }
6536 }{
6537     \ifx\Gin@ewidth\@empty
6538         \ifx\Gin@mhrepos\@empty
6539             \mhgraphics[width=\slidewidth,#1]{#2}
6540         \else
6541             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6542         \fi
6543         \ifx\Gin@mhrepos\@empty
6544             \mhgraphics[#1]{#2}
6545         \else
6546             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6547         \fi
6548     \fi% Gin@ewidth empty
6549 }
6550 \end{center}
6551 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6552 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6553 }
6554 } % ifmks@sty@frameimages

```

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

39.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

6555 \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

```

6556 \AddToHook{begindocument}{
6557     \definecolor{green}{rgb}{0,.5,0}
6558     \definecolor{purple}{cmyk}{.3,1,0,.17}
6559 }

```

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`.

```

6560 % \def\STpresent#1{\textcolor{blue}{#1}}
6561 \def\defemph#1{\textcolor{magenta}{#1}}
6562 \def\symrefemph#1{\textcolor{cyan}{#1}}
6563 \def\compemph#1{\textcolor{blue}{#1}}
6564 \def\titleemph#1{\textcolor{blue}{#1}}
6565 \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.

```

6566 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6567 \def\smalltextwarning{
6568   \pgfuseimage{miko@small@dbend}
6569   \xspace
6570 }
6571 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6572 \newrobustcmd\textwarning{
6573   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6574   \xspace
6575 }
6576 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6577 \newrobustcmd\bigtextwarning{
6578   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6579   \xspace
6580 }

```

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

```

6581 \newrobustcmd\putgraphicsat[3]{
6582   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6583 }
6584 \newrobustcmd\putat[2]{
6585   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6586 }

```

39.6 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

6587 \bool_if:NT \c__notesslides_sectocframes_bool {
6588   \str_if_eq:VnTF \__notesslidestopsect{part}{
6589     \newcounter{chapter}\counterwithin*{section}{chapter}
6590   }{
6591     \str_if_eq:VnT \__notesslidestopsect{chapter}{
6592       \newcounter{chapter}\counterwithin*{section}{chapter}
6593     }
6594   }
6595 }

```

`\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
6596 \def\part@prefix{}
6597 \@ifpackageloaded{document-structure}{}{
6598   \str_case:VnF \__notesslidestopsect {
6599     {part}{
6600       \int_set:Nn \l_document_structure_section_level_int {0}
6601       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6602     \def\part@prefix{\arabic{chapter}.}
6603   }
6604   {chapter}{
6605     \int_set:Nn \l_document_structure_section_level_int {1}
6606     \def\thesection{\arabic{chapter}.\arabic{section}}
6607     \def\part@prefix{\arabic{chapter}.}
6608   }
6609 }{
6610   \int_set:Nn \l_document_structure_section_level_int {2}
6611   \def\part@prefix{}
6612 }
6613 }
6614
6615 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `omgroup` environment that choses the L^AT_EX sectioning macros according to `\section@level`.

sfragment

```

6616 \renewenvironment{sfragment}[2][]{
6617   \_document_structure_omgroup_args:n { #1 }
6618   \int_incr:N \l_document_structure_section_level_int
6619   \bool_if:NT \c__notesslides_sectocframes_bool {
6620     \stepcounter{slide}
6621     \begin{frame}[noframenumbering]
6622     \vfill\Large\centering
6623     \red{
6624       \ifcase\l_document_structure_section_level_int\or
6625         \stepcounter{part}
6626         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6627         \def\currentsectionlevel{\omdoc@part@kw}
6628       \or
6629         \stepcounter{chapter}
6630         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6631         \def\currentsectionlevel{\omdoc@chapter@kw}
6632       \or
6633         \stepcounter{section}
6634         \def\_notesslideslabel{\part@prefix\arabic{section}}
6635         \def\currentsectionlevel{\omdoc@section@kw}
6636       \or
6637         \stepcounter{subsection}
6638         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6639         \def\currentsectionlevel{\omdoc@subsection@kw}
6640       \or
6641         \stepcounter{subsubsection}
6642         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6643         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6644       \or
6645         \stepcounter{paragraph}
6646         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6647         \def\currentsectionlevel{\omdoc@paragraph@kw}
6648       \else
6649         \def\_notesslideslabel{}

```



```

6650         \def\currentsectionlevel{\omdoc@paragraph@kw}
6651         \fi% end ifcase
6652         \_notesslideslabel%\sref@label@id\_notesslideslabel
6653         \quad #2%
6654     }%
6655     \vfill%
6656     \end{frame}%
6657 }
6658 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6659     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6660 }
6661 }{}
6662 }

```

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

```

6663 \def\inserttheorembodyfont{\normalfont}
6664 %\bool_if:NF \c__notesslides_notes_bool {
6665 % \defbeamertemplate{theorem begin}{miko}
6666 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6667 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6668 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6669 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6670 % \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.

```

6671 % \expandafter\def\csname Parent2\endcsname{}
6672 %}
6673
6674 \AddToHook{begindocument}{% this does not work for some reason
6675     \setbeamertemplate{theorems}[ams style]
6676 }
6677 \bool_if:NT \c__notesslides_notes_bool {
6678     \renewenvironment{columns}[1][{}]{%
6679         \par\noindent%
6680         \begin{minipage}%
6681             \slidewidth\centering\leavevmode%
6682     }{}%
6683     \end{minipage}\par\noindent%
6684 }%
6685 \newsavebox\columnbox%
6686 \renewenvironment<>{column}[2][{}]{%
6687     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6688 }{}%
6689     \end{minipage}\end{lrbox}\usebox\columnbox%
6690 }%
6691 }
6692 \bool_if:NTF \c__notesslides_noproblems_bool {
6693     \newenvironment{problems}{}{}
6694 }{
6695     \excludecomment{problems}
6696 }

```

39.7 Excursions

`\excursion` The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```

6697 \gdef\printexcursions{}
6698 \newcommand\excursionref[2]{% label, text
6699   \bool_if:NT \c__notesslides_notes_bool {
6700     \begin{sparagraph}[title=Excursion]
6701       #2 \sref[fallback=the appendix]{#1}.
6702     \end{sparagraph}
6703   }
6704 }
6705 \newcommand\activate@excursion[2][]{
6706   \gappto\printexcursions{\inputref{#1}{#2}}
6707 }
6708 \newcommand\excursion[4][]{% repos, label, path, text
6709   \bool_if:NT \c__notesslides_notes_bool {
6710     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6711   }
6712 }

```

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

`\excursiongroup`

```

6713 \keys_define:nn{notesslides / excursiongroup }{
6714   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6715   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6716   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6717 }
6718 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6719   \tl_clear:N \l__notesslides_excursion_intro_tl
6720   \str_clear:N \l__notesslides_excursion_id_str
6721   \str_clear:N \l__notesslides_excursion_mhrepos_str
6722   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6723 }
6724 \newcommand\excursiongroup[1][]{
6725   \__notesslides_excursion_args:n{ #1 }
6726   \ifdefempty\printexcursions{}% only if there are excursions
6727   {\begin{note}
6728     \begin{sfragment}[#1]{Excursions}%
6729     \ifdefempty\l__notesslides_excursion_intro_tl{
6730       \inputref[\l__notesslides_excursion_mhrepos_str]{
6731         \l__notesslides_excursion_intro_tl
6732       }
6733     }
6734     \printexcursions%
6735     \end{sfragment}
6736   } \end{note}}
6737 }
6738 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6739 } \package

```

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

Chapter 40

The Implementation

40.1 Package Options

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

```
6740 <*package>
6741 <@@=problems>
6742 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6743 \RequirePackage{l3keys2e,stex}
6744
6745 \keys_define:nn { problem / pkg }{
6746   notes      .default:n    = { true },
6747   notes      .bool_set:N   = \c__problems_notes_bool,
6748   gnotes     .default:n    = { true },
6749   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6750   hints      .default:n    = { true },
6751   hints      .bool_set:N   = \c__problems_hints_bool,
6752   solutions  .default:n    = { true },
6753   solutions  .bool_set:N   = \c__problems_solutions_bool,
6754   pts        .default:n    = { true },
6755   pts        .bool_set:N   = \c__problems_pts_bool,
6756   min        .default:n    = { true },
6757   min        .bool_set:N   = \c__problems_min_bool,
6758   boxed      .default:n    = { true },
6759   boxed      .bool_set:N   = \c__problems_boxed_bool,
6760   unknown    .code:n       = {}
6761 }
6762 \newif\ifsolutions
6763
6764 \ProcessKeysOptions{ problem / pkg }
6765 \bool_if:NTF \c__problems_solutions_bool {
6766   \solutionstrue
6767 }{
6768   \solutionsfalse
6769 }
```

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

```
6770 \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.

```
6771 \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.

```
6772 \def\prob@problem@kw{Problem}
6773 \def\prob@solution@kw{Solution}
6774 \def\prob@hint@kw{Hint}
6775 \def\prob@note@kw{Note}
6776 \def\prob@gnote@kw{Grading}
6777 \def\prob@pt@kw{pt}
6778 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6779 \AddToHook{begindocument}{
6780   \ltx@ifpackageloaded{babel}{
6781     \makeatletter
6782     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6783     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6784       \input{problem-ngerman.ldf}
6785     }
6786     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6787       \input{problem-finnish.ldf}
6788     }
6789     \clist_if_in:NnT \l_tmpa_clist {french}{
6790       \input{problem-french.ldf}
6791     }
6792     \clist_if_in:NnT \l_tmpa_clist {russian}{
6793       \input{problem-russian.ldf}
6794     }
6795     \makeatother
6796   }{}
6797 }
```

40.2 Problems and Solutions

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

```
6798 \keys_define:nn{ problem / problem }{
6799   id      .str_set:x:N = \l__problems_prob_id_str,
6800   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6801   min     .tl_set:N    = \l__problems_prob_min_tl,
6802   title   .tl_set:N    = \l__problems_prob_title_tl,
6803   type    .tl_set:N    = \l__problems_prob_type_tl,
6804   refnum  .int_set:N   = \l__problems_prob_refnum_int
6805 }
6806 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6807 \str_clear:N \l__problems_prob_id_str
6808 \tl_clear:N \l__problems_prob_pts_tl
6809 \tl_clear:N \l__problems_prob_min_tl
6810 \tl_clear:N \l__problems_prob_title_tl
6811 \tl_clear:N \l__problems_prob_type_tl
6812 \int_zero_new:N \l__problems_prob_refnum_int
6813 \keys_set:nn { problem / problem }{ #1 }
6814 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6815   \let\l__problems_prob_refnum_int\undefined
6816 }
6817 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6818 \newcounter{problem}
6819 \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.

```

6820 \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

```

6821 \newcommand\prob@number{
6822   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6823     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6824   }{
6825     \int_if_exist:NTF \l__problems_prob_refnum_int {
6826       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6827     }{
6828       \prob@label\theproblem
6829     }
6830   }
6831 }

```

(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.

```

6832 \newcommand\prob@title[3]{%
6833   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6834     #2 \l__problems_inclprob_title_tl #3
6835   }{
6836     \tl_if_exist:NTF \l__problems_prob_title_tl {
6837       #2 \l__problems_prob_title_tl #3
6838     }{
6839       #1
6840     }
6841   }
6842 }

```

(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.

```

6843 \def\prob@heading{
6844   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6845   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6846 }

```

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

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`sproblem`

```

6847 \newenvironment{sproblem}[1][]{
6848   \__problems_prob_args:n{#1}%\sref@target%
6849   \@in@omtexttrue% we are in a statement (for inline definitions)
6850   \stepcounter{problem}\record@problem
6851   \def\current@section@level{\prob@problem@kw}
6852   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6853     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6854   }{
6855     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6856   }
6857   \str_if_exist:NTF \l__problems_inclprob_id_str {
6858     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6859   }{
6860     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6861   }
6862
6863
6864   \clist_set:No \l_tmpa_clist \sproblemtype
6865   \tl_clear:N \l_tmpa_tl
6866   \clist_map_inline:Nn \l_tmpa_clist {
6867     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6868       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6869     }
6870   }
6871   \tl_if_empty:NTF \l_tmpa_tl {
6872     \__problems_sproblem_start:
6873   }{
6874     \l_tmpa_tl
6875   }
6876   \stex_ref_new_doc_target:n \sproblemid
6877 }{
6878   \clist_set:No \l_tmpa_clist \sproblemtype
6879   \tl_clear:N \l_tmpa_tl
6880   \clist_map_inline:Nn \l_tmpa_clist {
6881     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6882       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6883     }

```

```

6884 }
6885 \tl_if_empty:NTF \l_tmpa_tl {
6886   \__problems_sproblem_end:
6887 }{
6888   \l_tmpa_tl
6889 }
6890
6891
6892 \smallskip
6893 }
6894
6895
6896 \cs_new_protected:Nn \__problems_sproblem_start: {
6897   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6898 }
6899 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6900
6901 \newcommand\stexpatchproblem[3][] {
6902   \str_set:Nx \l_tmpa_str{ #1 }
6903   \str_if_empty:NTF \l_tmpa_str {
6904     \tl_set:Nn \__problems_sproblem_start: { #2 }
6905     \tl_set:Nn \__problems_sproblem_end: { #3 }
6906   }{
6907     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6908     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6909   }
6910 }
6911
6912
6913 \bool_if:NT \c__problems_boxed_bool {
6914   \surroundwithmdframed{problem}
6915 }

```

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

```

6916 \def\record@problem{
6917   \protected@write\@auxout{}
6918   {
6919     \string\@problem{\prob@number}
6920     {
6921       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6922         \l__problems_inclprob_pts_tl
6923       }{
6924         \l__problems_prob_pts_tl
6925       }
6926     }%
6927     {
6928       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6929         \l__problems_inclprob_min_tl
6930       }{
6931         \l__problems_prob_min_tl
6932       }
6933     }
6934   }
6935 }

```

(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).

```
6936 \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.

```
6937 \keys_define:nn { problem / solution }{
6938   id                .str_set_x:N = \l__problems_solution_id_str ,
6939   for               .tl_set:N    = \l__problems_solution_for_tl ,
6940   height            .dim_set:N   = \l__problems_solution_height_dim ,
6941   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6942   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
6943   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6944 }
6945 \cs_new_protected:Nn \__problems_solution_args:n {
6946   \str_clear:N \l__problems_solution_id_str
6947   \tl_clear:N \l__problems_solution_for_tl
6948   \tl_clear:N \l__problems_solution_srccite_tl
6949   \clist_clear:N \l__problems_solution_creators_clist
6950   \clist_clear:N \l__problems_solution_contributors_clist
6951   \dim_zero:N \l__problems_solution_height_dim
6952   \keys_set:nn { problem / solution }{ #1 }
6953 }
```

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

```
6954 \newcommand\@startsolution[1][{}]{
6955   \__problems_solution_args:n { #1 }
6956   \@in@omtexttrue% we are in a statement.
6957   \bool_if:NF \c__problems_boxed_bool { \hrule }
6958   \smallskip\noindent
6959   {\textbf\prob@solution@kw : \enspace}
6960   \begin{small}
6961   \def\current@section@level{\prob@solution@kw}
6962   \ignorespacesandpars
6963 }
```

\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.

```
6964 \newcommand\startsolutions{
6965   \specialcomment{solution}{\@startsolution}{
6966     \bool_if:NF \c__problems_boxed_bool {
6967       \hrule\medskip
6968     }
6969     \end{small}%
6970   }
6971   \bool_if:NT \c__problems_boxed_bool {
6972     \surroundwithmdframed{solution}
6973   }
6974 }
```


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

\stopsolutions

```
6975 \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.

```
6976 \ifsolutions
6977   \startsolutions
6978 \else
6979   \stopsolutions
6980 \fi
```

exnote

```
6981 \bool_if:NTF \c__problems_notes_bool {
6982   \newenvironment{exnote}[1][]{
6983     \par\smallskip\hrule\smallskip
6984     \noindent\textbf{\prob@note@kw : }\small
6985   }{
6986     \smallskip\hrule
6987   }
6988 }{
6989   \excludecomment{exnote}
6990 }
```

hint

```
6991 \bool_if:NTF \c__problems_notes_bool {
6992   \newenvironment{hint}[1][]{
6993     \par\smallskip\hrule\smallskip
6994     \noindent\textbf{\prob@hint@kw :~ }\small
6995   }{
6996     \smallskip\hrule
6997   }
6998 \newenvironment{exhint}[1][]{
6999   \par\smallskip\hrule\smallskip
7000   \noindent\textbf{\prob@hint@kw :~ }\small
7001 }{
7002   \smallskip\hrule
7003 }
7004 }{
7005   \excludecomment{hint}
7006   \excludecomment{exhint}
7007 }
```

gnote

```
7008 \bool_if:NTF \c__problems_notes_bool {
7009   \newenvironment{gnote}[1][]{
7010     \par\smallskip\hrule\smallskip
7011     \noindent\textbf{\prob@gnote@kw : }\small
7012   }{
7013     \smallskip\hrule
7014   }
7015 }{
7016   \excludecomment{gnote}
7017 }
```

40.3 Multiple Choice Blocks

EdN:22

mcb 22

```

7018 \newenvironment{mcb}{
7019   \begin{enumerate}
7020 }{
7021   \end{enumerate}
7022 }
```

we define the keys for the mcc macro

```

7023 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7024   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7025     \bool_set_true:N #1
7026   }{
7027     \bool_set_false:N #1
7028   }
7029 }
7030 \keys_define:nn { problem / mcc }{
7031   id          .str_set:x:N = \l__problems_mcc_id_str ,
7032   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7033   T           .default:n   = { true } ,
7034   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7035   F           .default:n   = { true } ,
7036   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7037   Ttext       .code:n      = {
7038     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7039   } ,
7040   Ftext       .code:n      = {
7041     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7042   }
7043 }
7044 \cs_new_protected:Nn \l__problems_mcc_args:n {
7045   \str_clear:N \l__problems_mcc_id_str
7046   \tl_clear:N \l__problems_mcc_feedback_tl
7047   \bool_set_true:N \l__problems_mcc_t_bool
7048   \bool_set_true:N \l__problems_mcc_f_bool
7049   \bool_set_true:N \l__problems_mcc_Ttext_bool
7050   \bool_set_false:N \l__problems_mcc_Ftext_bool
7051   \keys_set:nn { problem / mcc }{ #1 }
7052 }
```

\mcc

```

7053 \newcommand\mcc[2][] {
7054   \l__problems_mcc_args:n{ #1 }
7055   \item #2
7056   \ifsolutions
7057     \\\
7058     \bool_if:NT \l__problems_mcc_t_bool {
7059       % TODO!
7060       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7061     }
7062     \bool_if:NT \l__problems_mcc_f_bool {
```

²²EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7063      % TODO!
7064      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7065    }
7066    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7067      !
7068    }{
7069      \l__problems_mcc_feedback_tl
7070    }
7071    \fi
7072  } %solutions

```

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

40.4 Including Problems

\includeproblem The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

7073
7074 \keys_define:nn{ problem / inclproblem }{
7075   id      .str_set:N = \l__problems_inclprob_id_str,
7076   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7077   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7078   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7079   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7080   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7081   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7082 }
7083 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7084   \str_clear:N \l__problems_prob_id_str
7085   \tl_clear:N \l__problems_inclprob_pts_tl
7086   \tl_clear:N \l__problems_inclprob_min_tl
7087   \tl_clear:N \l__problems_inclprob_title_tl
7088   \tl_clear:N \l__problems_inclprob_type_tl
7089   \int_zero_new:N \l__problems_inclprob_refnum_int
7090   \str_clear:N \l__problems_inclprob_mhrepos_str
7091   \keys_set:nn { problem / inclproblem }{ #1 }
7092   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7093     \let\l__problems_inclprob_pts_tl\undefined
7094   }
7095   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7096     \let\l__problems_inclprob_min_tl\undefined
7097   }
7098   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7099     \let\l__problems_inclprob_title_tl\undefined
7100   }
7101   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7102     \let\l__problems_inclprob_type_tl\undefined
7103   }
7104   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7105     \let\l__problems_inclprob_refnum_int\undefined
7106   }
7107 }

```

```

7108
7109 \cs_new_protected:Nn \__problems_inclprob_clear: {
7110   \let\l__problems_inclprob_id_str\undefined
7111   \let\l__problems_inclprob_pts_tl\undefined
7112   \let\l__problems_inclprob_min_tl\undefined
7113   \let\l__problems_inclprob_title_tl\undefined
7114   \let\l__problems_inclprob_type_tl\undefined
7115   \let\l__problems_inclprob_refnum_int\undefined
7116   \let\l__problems_inclprob_mhrepos_str\undefined
7117 }
7118 \__problems_inclprob_clear:
7119
7120 \newcommand\includeproblem[2][ ]{
7121   \__problems_inclprob_args:n{ #1 }
7122   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7123     \input{#2}
7124   }{
7125     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7126       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7127     }
7128   }
7129   \__problems_inclprob_clear:
7130 }

```

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

40.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

7131 \AddToHook{enddocument}{
7132   \bool_if:NT \c__problems_pts_bool {
7133     \message{Total:~\arabic{pts}~points}
7134   }
7135   \bool_if:NT \c__problems_min_bool {
7136     \message{Total:~\arabic{min}~minutes}
7137   }
7138 }

```

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

```

7139 \def\pts#1{
7140   \bool_if:NT \c__problems_pts_bool {
7141     \marginpar{#1~\prob@pt@kw}
7142   }
7143 }
7144 \def\min#1{
7145   \bool_if:NT \c__problems_min_bool {
7146     \marginpar{#1~\prob@min@kw}
7147   }
7148 }

```

`\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.

```

7149 \newcounter{pts}
7150 \def\show@pts{
7151   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7152     \bool_if:NT \c__problems_pts_bool {
7153       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7154       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7155     }
7156   }{
7157     \tl_if_exist:NT \l__problems_prob_pts_tl {
7158       \bool_if:NT \c__problems_pts_bool {
7159         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7160         \addtocounter{pts}{\l__problems_prob_pts_tl}
7161       }
7162     }
7163   }
7164 }

```

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

`\show@min`

```

7165 \newcounter{min}
7166 \def\show@min{
7167   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7168     \bool_if:NT \c__problems_min_bool {
7169       \marginpar{\l__problems_inclprob_min_tl\ min}
7170       \addtocounter{min}{\l__problems_inclprob_min_tl}
7171     }
7172   }{
7173     \tl_if_exist:NT \l__problems_prob_min_tl {
7174       \bool_if:NT \c__problems_min_bool {
7175         \marginpar{\l__problems_prob_min_tl\ min}
7176         \addtocounter{min}{\l__problems_prob_min_tl}
7177       }
7178     }
7179   }
7180 }
7181 </package>

```

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

Chapter 41

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

41.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
7182 <@@=hwexam>
7183 <*cls>
7184 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7185 \RequirePackage{l3keys2e}
7186 \DeclareOption*{
7187   \PassOptionsToClass{\CurrentOption}{document-structure}
7188   \PassOptionsToPackage{\CurrentOption}{stex}
7189   \PassOptionsToPackage{\CurrentOption}{hwexam}
7190   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7191 }
7192 \ProcessOptions
```

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

```
7193 \LoadClass{document-structure}
7194 \RequirePackage{stex}
7195 \RequirePackage{hwexam}
7196 \RequirePackage{tikzinput}
7197 \RequirePackage{graphicx}
7198 \RequirePackage{a4wide}
7199 \RequirePackage{amssymb}
7200 \RequirePackage{amstext}
7201 \RequirePackage{amsmath}
```

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

```

7202 \newcommand\assig@default@type{\hwexam@assignment@kw}
7203 \def\document@hwexamtype{\assig@default@type}
7204 <@@=document_structure>
7205 \keys_define:nn { document-structure / document }{
7206 id .str_set_x:N = \c_document_structure_document_id_str,
7207 hwexamtype .tl_set:N = \document@hwexamtype
7208 }
7209 <@@=hwexam>
7210 </cls>

```

Chapter 42

Implementation: The hwexam Package

42.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
7211  $\langle *package \rangle$ 
7212  $\backslash ProvidesExplPackage\{hwexam\}\{2022/02/26\}\{3.0.1\}\{homework assignments and exams\}$ 
7213  $\backslash RequirePackage\{l3keys2e\}$ 
7214
7215  $\backslash newif\iftest\testfalse$ 
7216  $\backslash DeclareOption\{test\}\{\testtrue\}$ 
7217  $\backslash newif\ifmultiple\multiplefalse$ 
7218  $\backslash DeclareOption\{multiple\}\{\multipletrue\}$ 
7219  $\backslash DeclareOption*\{\backslash PassOptionsToPackage\{\CurrentOption\}\{problem\}\}$ 
7220  $\backslash ProcessOptions$ 
```

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

```
7221  $\backslash RequirePackage\{keyval\}\{1997/11/10\}$ 
7222  $\backslash RequirePackage\{problem\}$ 
```

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

```
7223  $\backslash newcommand\hwexam@assignment@kw\{Assignment\}$ 
7224  $\backslash newcommand\hwexam@given@kw\{Given\}$ 
7225  $\backslash newcommand\hwexam@due@kw\{Due\}$ 
7226  $\backslash newcommand\hwexam@testemptypage@kw\{This~page~was~intentionally~left~$ 
7227  $blank~for~extra~space\}$ 
7228  $\backslash def\hwexam@minutes@kw\{minutes\}$ 
7229  $\backslash newcommand\correction@probs@kw\{prob.\}$ 
7230  $\backslash newcommand\correction@pts@kw\{total\}$ 
7231  $\backslash newcommand\correction@reached@kw\{reached\}$ 
7232  $\backslash newcommand\correction@sum@kw\{Sum\}$ 
7233  $\backslash newcommand\correction@grade@kw\{grade\}$ 
7234  $\backslash newcommand\correction@forgrading@kw\{To~be~used~for~grading,~do~not~write~here\}$ 
```


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

For the other languages, we set up triggers

```

7235 \AddToHook{begindocument}{
7236 \ltx@ifpackageloaded{babel}{
7237 \makeatletter
7238 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7239 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7240 \input{hwexam-ngerman.ldf}
7241 }
7242 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7243 \input{hwexam-finnish.ldf}
7244 }
7245 \clist_if_in:NnT \l_tmpa_clist {french}{
7246 \input{hwexam-french.ldf}
7247 }
7248 \clist_if_in:NnT \l_tmpa_clist {russian}{
7249 \input{hwexam-russian.ldf}
7250 }
7251 \makeatother
7252 }{}
7253 }
7254

```

42.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

7255 \newcounter{assignment}
7256 \numberproblemsin{assignment}
7257 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7258 \keys_define:nn { hwexam / assignment } {
7259 id .str_set:N = \l__hwexam_assign_id_str,
7260 number .int_set:N = \l__hwexam_assign_number_int,
7261 title .tl_set:N = \l__hwexam_assign_title_tl,
7262 type .tl_set:N = \l__hwexam_assign_type_tl,
7263 given .tl_set:N = \l__hwexam_assign_given_tl,
7264 due .tl_set:N = \l__hwexam_assign_due_tl,
7265 loadmodules .code:n = {
7266 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7267 }
7268 }
7269 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7270 \str_clear:N \l__hwexam_assign_id_str
7271 \int_set:Nn \l__hwexam_assign_number_int {-1}
7272 \tl_clear:N \l__hwexam_assign_title_tl
7273 \tl_clear:N \l__hwexam_assign_type_tl
7274 \tl_clear:N \l__hwexam_assign_given_tl
7275 \tl_clear:N \l__hwexam_assign_due_tl
7276 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7277 \keys_set:nn { hwexam / assignment }{ #1 }
7278 }

```

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.

```

7279 \newcommand\given@due[2]{
7280 \bool_lazy_all:nF {
7281 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7282 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7283 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7284 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7285 }{ #1 }
7286
7287 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7288 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7289 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7290 }
7291 }{
7292 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7293 }
7294
7295 \bool_lazy_or:nnF {
7296 \bool_lazy_and_p:nn {
7297 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7298 }{
7299 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7300 }
7301 }{
7302 \bool_lazy_and_p:nn {
7303 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7304 }{
7305 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7306 }
7307 }{ ,~ }
7308
7309 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7310 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7311 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7312 }
7313 }{
7314 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7315 }
7316
7317 \bool_lazy_all:nF {
7318 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7319 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7320 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7321 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7322 }{ #2 }
7323 }

```

`\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.

```

7324 \newcommand\assignment@title[3]{
7325   \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
7326     \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7327       #1
7328     }{
7329       #2\l__hwexam_assign_title_tl#3
7330     }
7331   }{
7332     #2\l__hwexam_inclassassign_title_tl#3
7333   }
7334 }

```

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

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

```

7335 \newcommand\assignment@number{
7336   \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
7337     \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7338       \arabic{assignment}
7339     } {
7340       \int_use:N \l__hwexam_assign_number_int
7341     }
7342   }{
7343     \int_use:N \l__hwexam_inclassassign_number_int
7344   }
7345 }

```

(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.

```

7346 \newenvironment{assignment}[1][ ]{
7347   \__hwexam_assignment_args:n { #1 }
7348   %\sref@target
7349   \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7350     \global\stepcounter{assignment}
7351   }{
7352     \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7353   }
7354   \setcounter{problem}{0}
7355   \def\current@section@level{\document@hwexamtype}
7356   %\sref@label@id{\document@hwexamtype \thesection}
7357   \begin{@assignment}
7358 }{
7359   \end{@assignment}
7360 }

```

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

```

7361 \def\ass@title{
7362 \protect\document@hwexamtype~\arabic{assignment}
7363 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7364 }
7365 \ifmultiple
7366 \newenvironment{@assignment}{
7367 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7368 \begin{sfragment}[loadmodules]{\ass@title}
7369 }{
7370 \begin{sfragment}{\ass@title}
7371 }
7372 }{
7373 \end{sfragment}
7374 }

```

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

```

7375 \else
7376 \newenvironment{@assignment}{
7377 \begin{center}\bf
7378 \Large@title\strut\
7379 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7380 \large\given@due{--;\}{}{;\}{}
7381 \end{center}
7382 }{}
7383 \fi% multiple

```

42.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

7384 \keys_define:nn { hwexam / inclassignment } {
7385 %id .str_set_x:N = \l__hwexam_assign_id_str,
7386 number .int_set:N = \l__hwexam_inclassign_number_int,
7387 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7388 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7389 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7390 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7391 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7392 }
7393 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7394 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7395 \tl_clear:N \l__hwexam_inclassign_title_tl
7396 \tl_clear:N \l__hwexam_inclassign_type_tl
7397 \tl_clear:N \l__hwexam_inclassign_given_tl
7398 \tl_clear:N \l__hwexam_inclassign_due_tl
7399 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7400 \keys_set:nn { hwexam / inclassignment }{ #1 }
7401 }
7402 \__hwexam_inclassignment_args:n {}
7403
7404 \newcommand\inputassignment[2][{}]{

```

```

7405 \_hwexam_inclassnment_args:n { #1 }
7406 \str_if_empty:NTF \l__hwexam_inclassnign_mhrepos_str {
7407 \input{#2}
7408 }{
7409 \stex_in_repository:nn{\l__hwexam_inclassnign_mhrepos_str}{
7410 \input{\mhp{path}\l__hwexam_inclassnign_mhrepos_str}{#2}}
7411 }
7412 }
7413 \_hwexam_inclassnment_args:n {}
7414 }
7415 \newcommand\includeassignment[2][ ]{
7416 \newpage
7417 \inputassignment[#1]{#2}
7418 }

```

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

42.4 Typesetting Exams

\quizheading

```

7419 \ExplSyntaxOff
7420 \newcommand\quizheading[1]{%
7421 \def\@tas{#1}%
7422 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7423 \ifx\@tas\@empty\else%
7424 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7425 \fi%
7426 }
7427 \ExplSyntaxOn

```

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

\testheading

```

7428
7429 \def\hwexamheader{\input{hwexam-default.header}}
7430
7431 \def\hwexamminutes{
7432 \tl_if_empty:NTF \testheading@duration {
7433 {\testheading@min}~\hwexam@minutes@kw
7434 }{
7435 \testheading@duration
7436 }
7437 }
7438
7439 \keys_define:nn { hwexam / testheading } {
7440 min .tl_set:N = \testheading@min,
7441 duration .tl_set:N = \testheading@duration,
7442 reqpts .tl_set:N = \testheading@reqpts,
7443 tools .tl_set:N = \testheading@tools
7444 }
7445 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7446 \tl_clear:N \testheading@min
7447 \tl_clear:N \testheading@duration

```

```

7448 \tl_clear:N \testheading@reqpts
7449 \tl_clear:N \testheading@tools
7450 \keys_set:nn { hwexam / testheading }{ #1 }
7451 }
7452 \newenvironment{testheading}[1][]{
7453   \_hwexam_testheading_args:n{ #1 }
7454   \newcount\check@time\check@time=\testheading@min
7455   \advance\check@time by -\theassignment@totalmin
7456   \newif\if@bonuspoints
7457   \tl_if_empty:NTF \testheading@reqpts {
7458     \@bonuspointsfalse
7459   }{
7460     \newcount\bonus@pts
7461     \bonus@pts=\theassignment@totalpts
7462     \advance\bonus@pts by -\testheading@reqpts
7463     \edef\bonus@pts{\the\bonus@pts}
7464     \@bonuspointstrue
7465   }
7466   \edef\check@time{\the\check@time}
7467
7468   \makeatletter\hwexamheader\makeatother
7469 }{
7470   \newpage
7471 }

```

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

\testspace

```

7472 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7473 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7474 \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.

```

7475 <@=problems>
7476 \renewcommand\@problem[3]{
7477   \stepcounter{assignment@probs}
7478   \def\__problemspts{#2}
7479   \ifx\__problemspts\@empty\else
7480     \addtocounter{assignment@totalpts}{#2}
7481   \fi
7482   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7483   \xdef\correction@probs{\correction@probs & #1}%
7484   \xdef\correction@pts{\correction@pts & #2}
7485   \xdef\correction@reached{\correction@reached &}

```

```

7486 }
7487 <@@=hwexam>

```

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

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

```

7488 \newcounter{assignment@probs}
7489 \newcounter{assignment@totalpts}
7490 \newcounter{assignment@totalmin}
7491 \def\correction@probs{\correction@probs@kw}
7492 \def\correction@pts{\correction@pts@kw}
7493 \def\correction@reached{\correction@reached@kw}
7494 \stepcounter{assignment@probs}
7495 \newcommand\correction@table{
7496 \resizebox{\textwidth}{!}{%
7497 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7498 &\multicolumn{\theassignment@probs}{c|}|%|
7499 {\footnotesize\correction@forgrading@kw} &\\ \hline
7500 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7501 \correction@pts & \theassignment@totalpts & \\ \hline
7502 \correction@reached & & \[.7cm]\hline
7503 \end{tabular}}
7504 </package>

```

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

42.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```