

The sTeX3 Package *

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

2022-03-03

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 2022-03-03)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
3	Using sTeX	6
4	sTeX Archives	7
4.1	The Local MathHub-Directory	7
4.2	The Structure of sTeX Archives	7
4.3	MANIFEST.MF-Files	8
5	Creating New Modules and Symbols	9
5.1	Advanced Structuring Mechanisms	9
5.2	Primitive Symbols (The sTeX Metatheory)	11
6	sTeX Statements (Definitions, Theorems, Examples, ...)	12
7	Additional Packages	13
7.1	Modular Document Structuring	13
7.2	Slides and Course Notes	13
7.3	Homework, Problems and Exams	13
8	Stuff	14
8.1	Modules	14
8.1.1	Semantic Macros and Notations	14
	Other Argument Types	16
	Precedences	18
8.1.2	Archives and Imports	18
	Namespaces	18
	Paths in Import-Statements	19
II	Documentation	20
9	sTeX-Basics	21
9.1	Macros and Environments	21
9.1.1	HTML Annotations	21
9.1.2	Babel Languages	22
9.1.3	Auxiliary Methods	22

10	<code>sTeX</code>-MathHub	23
10.1	Macros and Environments	23
10.1.1	Files, Paths, URIs	23
10.1.2	MathHub Archives	24
10.1.3	Using Content in Archives	25
11	<code>sTeX</code>-References	26
11.1	Macros and Environments	26
11.1.1	Setting Reference Targets	26
11.1.2	Using References	27
12	<code>sTeX</code>-Modules	28
12.1	Macros and Environments	28
12.1.1	The <code>smodule</code> environment	30
13	<code>sTeX</code>-Module Inheritance	32
13.1	Macros and Environments	32
13.1.1	SMS Mode	32
13.1.2	Imports and Inheritance	33
14	<code>sTeX</code>-Symbols	35
14.1	Macros and Environments	35
15	<code>sTeX</code>-Terms	37
15.1	Macros and Environments	37
16	<code>sTeX</code>-Structural Features	39
16.1	Macros and Environments	39
16.1.1	Structures	39
17	<code>sTeX</code>-Statements	40
17.1	Macros and Environments	40
18	<code>sTeX</code>-Proofs: Structural Markup for Proofs	41
18.1	Introduction	43
18.2	The User Interface	44
18.2.1	Package Options	44
18.2.2	Proofs and Proof steps	44
18.2.3	Justifications	44
18.2.4	Proof Structure	46
18.2.5	Proof End Markers	46
18.2.6	Configuration of the Presentation	46
18.3	Limitations	47
19	<code>sTeX</code>-Metatheory	48
19.1	Symbols	48
III	Extensions	49

20	Tikzinput	50
20.1	Macros and Environments	50
21	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	51
21.1	Introduction	51
21.2	The User Interface	52
21.2.1	Package and Class Options	52
21.2.2	Document Structure	52
21.2.3	Ignoring Inputs	54
21.2.4	Structure Sharing	54
21.2.5	Global Variables	54
21.2.6	Colors	55
21.3	Limitations	55
22	NotesSlides – Slides and Course Notes	56
22.1	Introduction	56
22.2	The User Interface	56
22.2.1	Package Options	56
22.2.2	Notes and Slides	57
22.2.3	Header and Footer Lines of the Slides	58
22.2.4	Frame Images	58
22.2.5	Colors and Highlighting	59
22.2.6	Front Matter, Titles, etc.	59
22.2.7	Excursions	59
22.2.8	Miscellaneous	60
22.3	Limitations	60
23	problem.sty: An Infrastructure for formatting Problems	61
23.1	Introduction	61
23.2	The User Interface	61
23.2.1	Package Options	61
23.2.2	Problems and Solutions	62
23.2.3	Multiple Choice Blocks	63
23.2.4	Including Problems	63
23.2.5	Reporting Metadata	63
23.3	Limitations	63
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	65
24.1	Introduction	66
24.2	The User Interface	66
24.2.1	Package and Class Options	66
24.2.2	Assignments	66
24.2.3	Typesetting Exams	66
24.2.4	Including Assignments	67
24.3	Limitations	67
IV	Implementation	69

25	STeX-Basics Implementation	70
25.1	The STeXDocument Class	70
25.2	Preliminaries	70
25.3	Messages and logging	71
25.4	HTML Annotations	72
25.5	Babel Languages	75
25.6	Auxiliary Methods	76
26	STeX-MathHub Implementation	77
26.1	Generic Path Handling	77
26.2	PWD and kpsewhich	79
26.3	File Hooks and Tracking	80
26.4	MathHub Repositories	81
26.5	Using Content in Archives	85
27	STeX-References Implementation	90
27.1	Document URIs and URLs	90
27.2	Setting Reference Targets	92
27.3	Using References	94
28	STeX-Modules Implementation	97
28.1	The smodule environment	101
28.2	Invoking modules	106
29	STeX-Module Inheritance Implementation	108
29.1	SMS Mode	108
29.2	Inheritance	111
30	STeX-Symbols Implementation	116
30.1	Symbol Declarations	116
30.2	Notations	122
30.3	Variables	132
31	STeX-Terms Implementation	138
31.1	Symbol Invocations	138
31.2	Terms	145
31.3	Notation Components	149
31.4	Variables	151
31.5	Sequences	153
32	STeX-Structural Features Implementation	154
32.1	Imports with modification	155
32.2	The feature environment	161
32.3	Structure	162
33	STeX-Statements Implementation	170
33.1	Definitions	170
33.2	Assertions	175
33.3	Examples	178
33.4	Logical Paragraphs	181

34 The Implementation	186
34.1 Package Options	186
34.2 Proofs	186
34.3 Justifications	197
35 \LaTeX-Others Implementation	199
36 \LaTeX-Metatheory Implementation	200
37 Tikzinput Implementation	203
38 document-structure.sty Implementation	205
38.1 The document-structure Class	205
38.2 Class Options	205
38.3 Beefing up the <code>document</code> environment	206
38.4 Implementation: document-structure Package	206
38.5 Package Options	206
38.6 Document Structure	208
38.7 Front and Backmatter	211
38.8 Global Variables	213
39 NotesSlides – Implementation	214
39.1 Class and Package Options	214
39.2 Notes and Slides	216
39.3 Header and Footer Lines	220
39.4 Frame Images	221
39.5 Colors and Highlighting	222
39.6 Sectioning	223
39.7 Excursions	226
40 The Implementation	227
40.1 Package Options	227
40.2 Problems and Solutions	228
40.3 Multiple Choice Blocks	234
40.4 Including Problems	235
40.5 Reporting Metadata	236
41 Implementation: The hwexam Class	238
41.1 Class Options	238
42 Implementation: The hwexam Package	240
42.1 Package Options	240
42.2 Assignments	241
42.3 Including Assignments	244
42.4 Typesetting Exams	245
42.5 Leftovers	247

Part I

Manual



Implementation Details

$\hookrightarrow M \rightarrow$

$\dashv M \dashv$ MMT/OMDoc Info

$\sim T \rightsquigarrow$

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

2.2 A First $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ Document

Having set everything up, we can write a first $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder.

The document we will consider is the following:

```

1 \documentclass{article}
2 \usepackage{stex}
3 \usepackage{xcolor}
4 \def\compemph#1{\textcolor{blue}{#1}}
5
6 \begin{document}
7   \usemodule[smglom/calculus]{series}
8   \usemodule[smglom/arithmetics]{realarith}
9
10  The \symref{series}{series}  $\sum_{n=1}^{\infty} \frac{1}{2^n}$ 
11     \realdive[frac]{1}{2}
12     \realpower{2}{n}
13  }
14  }$ \symref{converges}{converges} towards 1$.
15
16 \end{document}
```

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

$\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitiesum` later on.

Analogously, `\usemodule[smglom/arithmetics]{realarith}` opens the file `realarith.en.tex` in the `.../smglom/arithmetics/source-folder` and makes its contents available, e.g. `\realdive` and `\realpower`.

³EdNOTE: somewhere later

`\symref`
`\symname`

The command `\symref{symbolname}{text}` marks the `text` in the second argument as representing the `symbolname` in the first argument – which is why the word “series” is set in boldface. In the pdf, this is all that happens. In the `xhtml` (which we will investigate shortly) however, we will note that the word “series” is now annotated with the full URI of the symbol denoting the *mathematical concept of a series*. In other words, the word is associated with an unambiguous semantics.

Notably, in both cases above (*series* and *converges*) the text that *references* the symbol and the name of the symbol are identical. Since this occurs quite often, the shorthand `\symname{converges}` would have worked as well, where `\symname{foo-bar}` behaves exactly like `\symref{foo-bar}{foo bar}` - i.e. the text is simply the name of the symbol with “-” replaced by a space.

`\importmodule`

If you investigated the contents of the imported modules (`realarith` and `series`) more closely, you’ll note that none of them contain a symbol “converges”. Yet, we can use `\symref` to refer to “converges”. That is because the symbol `converges` is found in `smglom/calculus/source/sequenceConvergence.en.tex`, and `series.en.tex` contains the line `\importmodule{sequenceConvergence}`. The `\importmodule`-statement makes the module referenced available to all documents that include the current module. As such, a “current module” has to exist for `\importmodule` to work, which is why the command is only allowed within a `module-environment`.

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

Chapter 3

Using \LaTeX

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

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

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

sms ($\langle\textit{boolean}\rangle$) use *persisted* mode (not yet implemented).

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

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

TODO: [terms documentation](#)

TODO: [references documentation](#)

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     \renamedecl[name=universe]{universe}{runiverse}
4     \renamedecl[name=plus]{operation}{rplus}
5     \renamedecl[name=zero]{unit}{rzero}
6     \renamedecl[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    \renamedecl[name=times]{operation}{rtimes}
14    \renamedecl[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:

$$\text{\syndef}{forevery}[args=bi]{\forall \#1.\; \#2}$$

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. $\int \int \int f(x,y,z) dx dy dz$?

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 ’s 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:

 $\backslash\text{c_stex_module_}\langle <URI> \rangle_\text{prop}$

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.

 $\backslash\text{c_stex_module_}\langle <URI> \rangle_\text{code}$

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

 $\backslash\text{c_stex_module_}\langle <URI> \rangle_\text{constants}$

The names of all constants declared in the module

 $\backslash\text{c_stex_module_}\langle <URI> \rangle_\text{constants}$

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code>	<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>	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>	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>	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>	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>	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>	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>	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>	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>	Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.
<hr/> <hr/> <code>\stex_get_symbol:n</code>	Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...
<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

TeX-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}
\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.barderviv,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[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets \LaTeX ML generate the correct reference.

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

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{\langle vname \rangle}{\langle text \rangle}` to set the global variable `\langle vname \rangle` to `\langle text \rangle` and `\useSGvar{\langle vname \rangle}` to reference it.

`\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 `noframenumbers` 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 — 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` 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.

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.

Name: _____ Matriculation Number: _____

2022-03-03

Write the solutions to the sheet.

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

[illegible]

Example 8: A generated test heading.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{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   \ifundefined{texorpdfstring}%
36   {\let\texorpdfstring\@firstoftwo}%
37   {}%
38   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
39 }
40 \def\sTeX{\stex}

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 21.)

25.3 Messages and logging

```

41 <@@=stex_log>

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

```

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

```

52 \cs_new_protected:Nn \stex_debug:nn {
53   \clist_if_in:NnTF \c_stex_debug_clist { all } {
54     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
55       \\Debug~#1:~#2\\
56     }
57     \msg_none:nn{stex}{debug / #1}
58   }{
59     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
60       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
61         \\Debug~#1:~#2\\
62       }
63       \msg_none:nn{stex}{debug / #1}
64     }
65   }
66 }

```

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

Redirecting messages:

```

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

```

25.4 HTML Annotations

```

76 <@=stex_annotate>
77 \RequirePackage{rustex}

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

79 \ifcsname if@latexml\endcsname\else
80   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
81 \fi

```

(End definition for `\if@latexml`. This function is documented on page 21.)

`\latexml_if_p:`

`\latexml_if:TF`

```

82 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
83   \if@latexml
84     \prg_return_true:
85   \else:
86     \prg_return_false:
87   \fi:
88 }

```

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

`\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.

```

89 \tl_new:N \l__stex_annotate_arg_tl
90 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
91   \rustex_if:TF {
92     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
93   }{-}
94 }

```

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

`_stex_annotate_checkempty:n`

```

95 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
96   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
97   \tl_if_empty:NT \l__stex_annotate_arg_tl {
98     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
99   }
100 }

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

101 \bool_new:N \_stex_html_do_output_bool
102 \bool_set_true:N \_stex_html_do_output_bool
103
104 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
105   \bool_if:nTF \_stex_html_do_output_bool
106     \prg_return_true: \prg_return_false:
107 }

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

108 \cs_new_protected:Nn \stex_suppress_html:n {
109   \exp_args:Nne \use:nn {
110     \bool_set_false:N \_stex_html_do_output_bool
111     #1
112   }{
113     \stex_if_do_html:T {
114       \bool_set_true:N \_stex_html_do_output_bool
115     }
116   }
117 }

```

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

`\stex_annotate:nnx`

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`

```

118 \rustex_if:TF{
119   \cs_new_protected:Nn \stex_annotate:nnn {
120     \_stex_annotate_checkempty:n { #3 }
121     \rustex_annotate_HTML:nn {
122       property="stex:#1" ~
123       resource="#2"
124     } {
125       \mode_if_vertical:TF{
126         \tl_use:N \l__stex_annotate_arg_tl\par
127       }{
128         \tl_use:N \l__stex_annotate_arg_tl
129       }
130     }
131   }
132   \cs_new_protected:Nn \stex_annotate_invisible:n {

```

```

133 \__stex_annotate_checkempty:n { #1 }
134 \rustex_annotate_HTML:nn {
135   stex:visible="false" ~
136   style:display="none"
137 } {
138   \mode_if_vertical:TF{
139     \tl_use:N \l__stex_annotate_arg_tl\par
140   }{
141     \tl_use:N \l__stex_annotate_arg_tl
142   }
143 }
144 }
145 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
146   \__stex_annotate_checkempty:n { #3 }
147   \rustex_annotate_HTML:nn {
148     property="stex:#1" ~
149     resource="#2" ~
150     stex:visible="false" ~
151     style:display="none"
152   } {
153     \mode_if_vertical:TF{
154       \tl_use:N \l__stex_annotate_arg_tl\par
155     }{
156       \tl_use:N \l__stex_annotate_arg_tl
157     }
158   }
159 }
160 \NewDocumentEnvironment{stex_annotate_env} { m m } {
161   \par
162   \rustex_annotate_HTML_begin:n {
163     property="stex:#1" ~
164     resource="#2"
165   }
166 }{
167   \par\rustex_annotate_HTML_end:
168 }
169 }{
170   \latexml_if:TF {
171     \cs_new_protected:Nn \stex_annotate:nnn {
172       \__stex_annotate_checkempty:n { #3 }
173       \mode_if_math:TF {
174         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
175           \tl_use:N \l__stex_annotate_arg_tl
176         }
177       }{
178         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
179           \tl_use:N \l__stex_annotate_arg_tl
180         }
181       }
182     }
183     \cs_new_protected:Nn \stex_annotate_invisible:n {
184       \__stex_annotate_checkempty:n { #1 }
185       \mode_if_math:TF {
186         \cs:w latexml@invisible@math\cs_end:{

```



```

187         \tl_use:N \l__stex_annotate_arg_tl
188     }
189 } {
190     \cs:w latexml@invisible@text\cs_end:{
191         \tl_use:N \l__stex_annotate_arg_tl
192     }
193 }
194 }
195 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
196     \__stex_annotate_checkempty:n { #3 }
197     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
198         \tl_use:N \l__stex_annotate_arg_tl
199     }
200 }
201 \NewDocumentEnvironment{stex_annotate_env} { m m } {
202     \par\begin{latexml@annotateenv}{#1}{#2}
203 }{
204     \par\end{latexml@annotateenv}
205 }
206 }{
207     \cs_new_protected:Nn \stex_annotate:nnn {#3}
208     \cs_new_protected:Nn \stex_annotate_invisible:n {}
209     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
210     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
211 }
212 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 22.)

25.5 Babel Languages

```

213 <@@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:
`\c_stex_language_abbrevs_prop`

```

214 \prop_const_from_keyval:Nn \c_stex_languages_prop {
215     en = english ,
216     de = ngerman ,
217     ar = arabic ,
218     bg = bulgarian ,
219     ru = russian ,
220     fi = finnish ,
221     ro = romanian ,
222     tr = turkish ,
223     fr = french
224 }
225
226 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
227     english = en ,
228     ngerman = de ,
229     arabic = ar ,
230     bulgarian = bg ,
231     russian = ru ,
232     finnish = fi ,

```

```

233   romanian = ro ,
234   turkish  = tr ,
235   french   = fr
236 }
237 % todo: chinese simplified (zhs)
238 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 22.)

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

```

239 \clist_if_empty:NF \c_stex_languages_clist {
240   \clist_clear:N \l_tmpa_clist
241   \clist_map_inline:Nn \c_stex_languages_clist {
242     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
243       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
244     } {
245       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
246     }
247   }
248   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
249   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
250 }

```

25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

251 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
252   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
253   \def#1{
254     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
255   }
256 }

```

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

`\stex_reactivate_macro:N`

```

257 \cs_new_protected:Nn \stex_reactivate_macro:N {
258   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
259 }

```

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

`\ignorespacesandpars`

```

260 \protected\def\ignorespacesandpars{
261   \begingroup\catcode13=10\relax
262   \@ifnextchar\par{
263     \endgroup\expandafter\ignorespacesandpars\@gobble
264   }{
265     \endgroup
266   }
267 }
268 \</package>

```

(End definition for `\ignorespacesandpars`. This function is documented on page 22.)

Chapter 26

STEX -MathHub Implementation

```
269 <*package>
270
271 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
272
273 <@@=stex_path>
274
275 Warnings and error messages
276 \msg_new:nnn{stex}{error/norepository}{
277   No~archive~#1~found~in~#2
278 }
279 \msg_new:nnn{stex}{error/notinarchive}{
280   Not~currently~in~an~archive,~but~\detokenize{#1}~
281   needs~one!
282 }
283 \msg_new:nnn{stex}{error/nofile}{
284   \detokenize{#1}~could~not~find~file~#2
285 }
286 \msg_new:nnn{stex}{error/twofiles}{
287   \detokenize{#1}~found~two~candidates~for~#2
288 }
```

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`

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

```

295     \seq_map_inline:Nn #1 {
296       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
297       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
298     }
299     \seq_set_eq:NN #1 \l_tmpa_tl
300   }
301   \stex_path_canonicalize:N #1
302 }
303 }
304

```

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

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

```

305 \cs_new_protected:Nn \stex_path_to_string:NN {
306   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
307 }
308
309 \cs_new:Nn \stex_path_to_string:N {
310   \seq_use:Nn #1 /
311 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 23.)

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

```

312 \str_const:Nn \c__stex_path_dot_str {.}
313 \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.

```

314 \cs_new_protected:Nn \stex_path_canonicalize:N {
315   \seq_if_empty:NF #1 {
316     \seq_clear:N \l_tmpa_seq
317     \seq_get_left:NN #1 \l_tmpa_tl
318     \str_if_empty:NT \l_tmpa_tl {
319       \seq_put_right:Nn \l_tmpa_seq {}
320     }
321     \seq_map_inline:Nn #1 {
322       \str_set:Nn \l_tmpa_tl { ##1 }
323       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
324         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
325           \seq_if_empty:NNTF \l_tmpa_seq {
326             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
327               \c__stex_path_up_str
328             }
329           }{
330             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
331             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
332               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
333                 \c__stex_path_up_str
334               }
335             }{

```

```

336         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
337     }
338 }
339 }{
340     \str_if_empty:NF \l_tmpa_tl {
341         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
342     }
343 }
344 }
345 }
346 \seq_gset_eq:NN #1 \l_tmpa_seq
347 }
348 }

```

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

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

```

349 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
350     \seq_if_empty:NTF #1 {
351         \prg_return_false:
352     }{
353         \seq_get_left:NN #1 \l_tmpa_tl
354         \sys_if_platform_windows:TF{
355             \str_if_in:NnTF \l_tmpa_tl {:}{
356                 \prg_return_true:
357             }{
358                 \prg_return_false:
359             }
360         }{
361             \str_if_empty:NTF \l_tmpa_tl {
362                 \prg_return_true:
363             }{
364                 \prg_return_false:
365             }
366         }
367     }
368 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 23.)

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 23.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

26.3 File Hooks and Tracking

```

387 <@@=stex_files>

```

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

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

`\g__stex_files_stack`

keeps track of file changes

```

388 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

389 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
390 \stex_path_from_string:Nn \c_stex_mainfile_seq
391   \c_stex_mainfile_str

```

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

`\g_stex_currentfile_seq`

```

392 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 24.)

`\stex_filestack_push:n`

```

393 \cs_new_protected:Nn \stex_filestack_push:n {
394   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
395   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
396     \stex_path_from_string:Nn\g_stex_currentfile_seq{
397       \c_stex_pwd_str/#1
398     }
399   }
400   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
401   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
402 }

```

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

`\stex_filestack_pop:`

```

403 \cs_new_protected:Nn \stex_filestack_pop: {
404   \seq_if_empty:NF\g__stex_files_stack{
405     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
406   }
407   \seq_if_empty:NTF\g__stex_files_stack{
408     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
409   }{
410     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
411     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
412   }
413 }
```

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

Hooks for the current file:

```

414 \AddToHook{file/before}{
415   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
416 }
417 \AddToHook{file/after}{
418   \stex_filestack_pop:
419 }
```

26.4 MathHub Repositories

420 `<@=stex_mathhub>`

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

`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

```

421 \str_if_empty:NTF\mathhub{
422   \sys_if_platform_windows:TF{
423     \begingroup\escapechar=-1\catcode'\=12
424     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
425     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
426     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
427   }{
428     \stex_kpsewhich:n{-var-value-MATHHUB}
429   }
430   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
431 }
432 \str_if_empty:NTF\c_stex_mathhub_str{
433   \msg_warning:nn{stex}{warning/nomathhub}
434 }{
435   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
436   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
437 }
438 }{
439   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
440   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
441     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
442       \c_stex_pwd_str/\mathhub
443     }
444   }
```

```

444 }
445 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
446 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
447 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 24.)

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

```

448 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
449   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
450     \str_set:Nx \l_tmpa_str { #1 }
451     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
452     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
453     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
454     \_stex_mathhub_find_manifest:N \l_tmpa_seq
455     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
456       \msg_error:nnxx{stex}{error/norepository}{#1}{
457         \stex_path_to_string:N \c_stex_mathhub_str
458       }
459     } {
460       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
461     }
462   }
463 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

464 \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`:

```

465 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
466   \seq_set_eq:NN\l_tmpa_seq #1
467   \bool_set_true:N\l_tmpa_bool
468   \bool_while_do:Nn \l_tmpa_bool {
469     \seq_if_empty:NTF \l_tmpa_seq {
470       \bool_set_false:N\l_tmpa_bool
471     } {
472       \file_if_exist:nTF{
473         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
474       } {
475         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
476         \bool_set_false:N\l_tmpa_bool
477       } {
478         \file_if_exist:nTF{
479           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
480         } {
481           \seq_put_right:Nn\l_tmpa_seq{META-INF}
482           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```



```

483         \bool_set_false:N\l_tmpa_bool
484     ){
485         \file_if_exist:nTF{
486             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
487         }{
488             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
489             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
490             \bool_set_false:N\l_tmpa_bool
491         }{
492             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
493         }
494     }
495 }
496 }
497 }
498 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
499 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```
500 \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:

```

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

```

```

529     {ns} {
530         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531         { ns } \l_tmpb_tl
532     }
533     {dependencies} {
534         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535         { deps } \l_tmpb_tl
536     }
537     }{}{}
538 }{}
539 }
540 \ior_close:N \c__stex_mathhub_manifest_ior
541 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

542 \cs_new_protected:Nn \stex_set_current_repository:n {
543     \stex_require_repository:n { #1 }
544     \prop_set_eq:Nc \l_stex_current_repository_prop {
545         c_stex_mathhub_#1_manifest_prop
546     }
547 }

```

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

`\stex_require_repository:n`

```

548 \cs_new_protected:Nn \stex_require_repository:n {
549     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
550         \stex_debug:nn{mathhub}{Opening~archive:~#1}
551         \_stex_mathhub_do_manifest:n { #1 }
552     }
553 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

554 %\prop_new:N \l_stex_current_repository_prop
555
556 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
557 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
558     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
559 } {
560     \_stex_mathhub_parse_manifest:n { main }
561     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
562     \l_tmpa_str
563     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
564     \c_stex_mathhub_main_manifest_prop
565     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
566     \stex_debug:nn{mathhub}{Current~repository:~
567         \prop_item:Nn \l_stex_current_repository_prop {id}
568     }
569 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 24.)

`\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.

```

570 \cs_new_protected:Nn \stex_in_repository:nn {
571   \str_set:Nx \l_tmpa_str { #1 }
572   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
573   \str_if_empty:NTF \l_tmpa_str {
574     \prop_if_exist:NTF \l_stex_current_repository_prop {
575       \stex_debug:nn{mathhub}{do-in~current~repository:~\prop_item:Nn \l_stex_current_reposi
576       \exp_args:Ne \l_tmpa_cs{
577         \prop_item:Nn \l_stex_current_repository_prop { id }
578       }
579     }{
580       \l_tmpa_cs{}
581     }
582   }{
583     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
584     \stex_require_repository:n \l_tmpa_str
585     \str_set:Nx \l_tmpa_str { #1 }
586     \exp_args:Nne \use:nn {
587       \stex_set_current_repository:n \l_tmpa_str
588       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
589     }{
590       \stex_debug:nn{mathhub}{switching~back~to:~
591       \prop_if_exist:NTF \l_stex_current_repository_prop {
592         \prop_item:Nn \l_stex_current_repository_prop { id }::~
593       \meaning\l_stex_current_repository_prop
594       }{
595         no~repository
596       }
597     }
598     \prop_if_exist:NTF \l_stex_current_repository_prop {
599       \stex_set_current_repository:n {
600         \prop_item:Nn \l_stex_current_repository_prop { id }
601       }
602     }{
603       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
604     }
605   }
606 }
607 }

```

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

26.5 Using Content in Archives

`\mhpath`

```

608 \def \mhpath #1 #2 {
609   \exp_args:Ne \tl_if_empty:nTF{#1}{
610     \c_stex_mathhub_str /
611     \prop_item:Nn \l_stex_current_repository_prop { id }
612     / source / #2
613   }{
614     \c_stex_mathhub_str / #1 / source / #2

```

```

615 }
616 }

```

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

`\inputref`
`\mhinput`

```

617 \newif \ifinputref \inputreffalse
618
619 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
620   \stex_in_repository:nn {#1} {
621     \ifinputref
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \else
624       \inputreftrue
625       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
626       \inputreffalse
627     \fi
628   }
629 }
630 \NewDocumentCommand \mhinput { 0{} m}{
631   \stex_mhinput:nn{ #1 }{ #2 }
632 }
633
634 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
635   \stex_in_repository:nn {#1} {
636     \bool_lazy_any:nTF {
637       {\rustex_if_p:}
638       {\latexml_if_p:}
639     } {
640       \str_clear:N \l_tmpa_str
641       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
642         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
643       }
644       \stex_annotate_invisible:nnn{inputref}{
645         \l_tmpa_str / #2
646       }{}
647     }{
648       \begingroup
649         \inputreftrue
650         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
651       \endgroup
652     }
653   }
654 }
655 \NewDocumentCommand \inputref { 0{} m}{
656   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
657 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 25.)

`\addmhbibresource`

```

658 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
659   \stex_in_repository:nn {#1} {
660     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
661   }

```

```

662 }
663 \newcommand\addmhbibresource[2][]{
664   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
665 }

```

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

\libinput

```

666 \cs_new_protected:Npn \libinput #1 {
667   \prop_if_exist:NF \l_stex_current_repository_prop {
668     \msg_error:nnn{stex}{error/notinarchive}\libinput
669   }
670   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
671     \msg_error:nnn{stex}{error/notinarchive}\libinput
672   }
673   \seq_clear:N \l__stex_mathhub_libinput_files_seq
674   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
675   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
676
677   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
678     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
679     \IfFileExists{ \l_tmpa_str }{
680       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
681     }{}
682     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
683     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
684   }
685
686   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
687   \IfFileExists{ \l_tmpa_str }{
688     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
689   }{}
690
691   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
692     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
693   }{
694     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
695       \input{ ##1 }
696     }
697   }
698 }

```

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

\libusepackage

```

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

```

```

709
710 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
711   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
712   \IfFileExists{ \l_tmpa_str.sty }{
713     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
714   }{}
715   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
716   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
717 }
718
719 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
720 \IfFileExists{ \l_tmpa_str.sty }{
721   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
722 }{}
723
724 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
725   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
726 }{
727   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
728     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
729       \usepackage[#1]{ #1 }
730     }
731   }{
732     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
733   }
734 }
735 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

736
737 \AddToHook{begindocument}{
738 \ltx@ifpackageloaded{graphicx}{
739   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
740   \newcommand\mhgraphics[2][]{%
741     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
742     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
743   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
744 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 25.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

745 \ltx@ifpackageloaded{listings}{
746   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
747   \newcommand\lstinputmhlisting[2][]{%
748     \def\lst@mhrepos{}\setkeys{lst}{#1}%
749     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
750   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
751 }{}
752 }
753
754 </package>

```

(End definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 25.)

Chapter 27

STEX -References Implementation

```
755 <*package>
756
757 %%%%%%%%%% references.dtx %%%%%%%%%%
758
759 <@@=stex_refs>
    Warnings and error messages
760
```

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

```
761 %\iow_new:N \c__stex_refs_refs_iow
762 \AddToHook{begindocument}{
763 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
764 }
765 \AddToHook{enddocument}{
766 % \iow_close:N \c__stex_refs_refs_iow
767 }
```

`\STEXreftitle`

```
768 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
769
770 \NewDocumentCommand \STEXreftitle { m } {
771 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
772 }
```

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

27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
773 \str_new:N \l_stex_current_docns_str
```

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

`\stex_get_document_uri:`

```

774 \cs_new_protected:Nn \stex_get_document_uri: {
775   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
776   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
777   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
778   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
779   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
780
781   \str_clear:N \l_tmpa_str
782   \prop_if_exist:NT \l_stex_current_repository_prop {
783     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
784       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
785     }
786   }
787
788   \str_if_empty:NTF \l_tmpa_str {
789     \str_set:Nx \l_stex_current_docns_str {
790       file:/\stex_path_to_string:N \l_tmpa_seq
791     }
792   }{
793     \bool_set_true:N \l_tmpa_bool
794     \bool_while_do:Nn \l_tmpa_bool {
795       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
796       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
797         {source} { \bool_set_false:N \l_tmpa_bool }
798       }{}{
799         \seq_if_empty:NT \l_tmpa_seq {
800           \bool_set_false:N \l_tmpa_bool
801         }
802       }
803     }
804
805     \seq_if_empty:NTF \l_tmpa_seq {
806       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
807     }{
808       \str_set:Nx \l_stex_current_docns_str {
809         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
810       }
811     }
812   }
813 }

```

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

`\l_stex_current_docurl_str`

```

814 \str_new:N \l_stex_current_docurl_str

```

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

`\stex_get_document_url:`

```

815 \cs_new_protected:Nn \stex_get_document_url: {
816   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
817   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
818   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str

```

```

819 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
820 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
821
822 \str_clear:N \l_tmpa_str
823 \prop_if_exist:NT \l_stex_current_repository_prop {
824   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
825     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
826       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
827     }
828   }
829 }
830
831 \str_if_empty:NTF \l_tmpa_str {
832   \str_set:Nx \l_stex_current_docurl_str {
833     file:/\stex_path_to_string:N \l_tmpa_seq
834   }
835 }{
836   \bool_set_true:N \l_tmpa_bool
837   \bool_while_do:Nn \l_tmpa_bool {
838     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
839     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
840       {source} { \bool_set_false:N \l_tmpa_bool }
841     }{}{
842       \seq_if_empty:NT \l_tmpa_seq {
843         \bool_set_false:N \l_tmpa_bool
844       }
845     }
846   }
847
848   \seq_if_empty:NTF \l_tmpa_seq {
849     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
850   }{
851     \str_set:Nx \l_stex_current_docurl_str {
852       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
853     }
854   }
855 }
856 }

```

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

27.2 Setting Reference Targets

```

857 \str_const:Nn \c__stex_refs_url_str{URL}
858 \str_const:Nn \c__stex_refs_ref_str{REF}
859 \str_new:N \l__stex_refs_curr_label_str
860 % @currentlabel -> number
861 % @currentlabelname -> title
862 % @currentHref -> name.number <- id of some kind
863 % \theH# -> \arabic{section}
864 % \the# -> number
865 % \hyper@makecurrent{#}
866 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

867 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
868   \stex_get_document_uri:
869   \str_clear:N \l__stex_refs_curr_label_str
870   \str_set:Nx \l_tmpa_str { #1 }
871   \str_if_empty:NT \l_tmpa_str {
872     \int_incr:N \l__stex_refs_unnamed_counter_int
873     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
874   }
875   \str_set:Nx \l__stex_refs_curr_label_str {
876     \l_stex_current_docns_str?\l_tmpa_str
877   }
878   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
879     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
880   }
881   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
882     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
883   }
884   \stex_if_smsmode:TF {
885     \stex_get_document_url:
886     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
887     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
888   }{
889     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
890     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
891     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
892     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
893   }
894 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 26.)

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

```

895 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
896   \str_set:Nn \l_tmpa_str {#1?#2}
897   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
898   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
899     \seq_new:c {g__stex_refs_labels_#2_seq}
900   }
901   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
902     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
903   }
904 }

```

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

```

905 \AtEndDocument{
906   \def\stexauxadddocref#1 #2 {}{}
907 }

```

`\stex_ref_new_sym_target:n`

```

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

```

```

913     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
914   }
915 }{
916   \str_if_empty:NF \l__stex_refs_curr_label_str {
917     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
918     \immediate\write\@auxout{
919       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
920       \l__stex_refs_curr_label_str
921     }
922   }
923 }
924 }
925 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 26.)

27.3 Using References

```

926 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

927
928 \keys_define:nn { stex / sref } {
929   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
930   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
931   pre           .tl_set:N = \l__stex_refs_pre_tl ,
932   post          .tl_set:N = \l__stex_refs_post_tl ,
933 }
934 \cs_new_protected:Nn \__stex_refs_args:n {
935   \tl_clear:N \l__stex_refs_linktext_tl
936   \tl_clear:N \l__stex_refs_fallback_tl
937   \tl_clear:N \l__stex_refs_pre_tl
938   \tl_clear:N \l__stex_refs_post_tl
939   \str_clear:N \l__stex_refs_repo_str
940   \keys_set:nn { stex / sref } { #1 }
941 }

```

The actual macro:

```

942 \NewDocumentCommand \sref { 0{} m}{
943   \__stex_refs_args:n { #1 }
944   \str_if_empty:NTF \l__stex_refs_indocument_str {
945     \str_set:Nx \l_tmpa_str { #2 }
946     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
947     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
948       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
949         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
950           \str_clear:N \l_tmpa_str
951         }
952       }{
953         \str_clear:N \l_tmpa_str
954       }
955     }{
956       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
957       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

958 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
959 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
960   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
961   \str_clear:N \l_tmpa_str
962   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
963     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
964       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
965     }{
966       \seq_map_break:n {
967         \str_set:Nn \l_tmpa_str { ##1 }
968       }
969     }
970   }
971 }{
972   \str_clear:N \l_tmpa_str
973 }
974 }
975 \str_if_empty:NTF \l_tmpa_str {
976   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
977 }{
978   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
979     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
980       \cs_if_exist:cTF{autoref}{
981         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
982       }{
983         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
984       }
985     }{
986       \ltx@ifpackageloaded{hyperref}{
987         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
988       }{
989         \l__stex_refs_linktext_tl
990       }
991     }
992   }{
993     \ltx@ifpackageloaded{hyperref}{
994       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
995     }{
996       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
997     }
998   }
999 }
1000 }{
1001   % TODO
1002 }
1003 }

```

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

`\srefsym`

```

1004 \NewDocumentCommand \srefsym { 0{} m }{
1005   \stex_get_symbol:n { #2 }
1006   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1007 }

```

```

1008
1009 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1010   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1011     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1012   }{
1013     \__stex_refs_args:n { #1 }
1014     \str_if_empty:NTF \l__stex_refs_indocument_str {
1015       \tl_if_exist:cTF{sref_sym_#2 _type}{
1016         % doc uri in \l_tmpb_str
1017         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1018         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1019           % reference
1020           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1021             \cs_if_exist:cTF{autoref}{
1022               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1023             }{
1024               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1025             }
1026           }{
1027             \ltx@ifpackageloaded{hyperref}{
1028               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1029             }{
1030               \l__stex_refs_linktext_tl
1031             }
1032           }
1033         }{
1034           % URL
1035           \ltx@ifpackageloaded{hyperref}{
1036             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1037           }{
1038             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1039           }
1040         }
1041       }{
1042         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1043       }
1044     }{
1045       % TODO
1046     }
1047   }
1048 }

```

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

\srefsymuri

```

1049 \cs_new_protected:Npn \srefsymuri #1 #2 {
1050   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1051 }

```

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

```

1052 </package>

```

Chapter 28

STEX -Modules Implementation

```
1053 <*package>
1054
1055 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1056
1057 <@@=stex_modules>
1058
1059     Warnings and error messages
1058 \msg_new:nnn{stex}{error/unknownmodule}{
1059     No~module~#1~found
1060 }
1061 \msg_new:nnn{stex}{error/syntax}{
1062     Syntax~error:~#1
1063 }
1064 \msg_new:nnn{stex}{error/siglanguage}{
1065     Module~#1~declares~signature~#2,~but~does~not~
1066     declare~its~language
1067 }
1068 \msg_new:nnn{stex}{warning/deprecated}{
1069     #1~is~deprecated;~please~use~#2~instead!
1070 }
1071
1072 \msg_new:nnn{stex}{error/conflictingmodules}{
1073     Conflicting~imports~for~module~#1
1074 }
1075
\l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
(End definition for \l_stex_current_module_str. This variable is documented on page 29.)

\l_stex_all_modules_seq Stores all available modules
1076 \seq_new:N \l_stex_all_modules_seq
1077
(End definition for \l_stex_all_modules_seq. This variable is documented on page 29.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1077 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1078   \str_if_empty:NTF \l_stex_current_module_str
1079   \prg_return_false: \prg_return_true:
1080 }

```

(End definition for \stex_if_in_module:TF. This function is documented on page 29.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1081 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1082   \prop_if_exist:cTF { c_stex_module_#1_prop }
1083   \prg_return_true: \prg_return_false:
1084 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 29.)

\stex_add_to_current_module:n Only allowed within modules:

```

\STEXexport
1085 \cs_new_protected:Nn \stex_add_to_current_module:n {
1086   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1087 }
1088 \cs_new_protected:Npn \STEXexport {
1089   \begingroup
1090   \newlinechar=-1\relax
1091   \endlinechar=-1\relax
1092   %\catcode'\ = 9\relax
1093   \expandafter\endgroup\__stex_modules_export:n
1094 }
1095 \cs_new_protected:Nn \__stex_modules_export:n {
1096   \ignorespaces #1
1097   \stex_add_to_current_module:n { \ignorespaces #1 }
1098   \stex_smsmode_do:
1099 }
1100 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 29.)

```

\stex_add_constant_to_current_module:n
1101 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1102   \str_set:Nx \l_tmpa_str { #1 }
1103   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1104 }

```

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page 29.)

```

\stex_add_import_to_current_module:n
1105 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1106   \str_set:Nx \l_tmpa_str { #1 }
1107   \exp_args:Nno
1108   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1109     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1110   }
1111 }

```


(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 29.)

`\stex_collect_imports:n`

```

1112 \cs_new_protected:Nn \stex_collect_imports:n {
1113   \seq_clear:N \l_stex_collect_imports_seq
1114   \__stex_modules_collect_imports:n {#1}
1115 }
1116 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1117   \seq_map_inline:cn {c_stex_module_#1_imports} {
1118     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1119       \__stex_modules_collect_imports:n { ##1 }
1120     }
1121   }
1122   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1123     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1124   }
1125 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 29.)

`\stex_do_up_to_module:n`

```

1126 \int_new:N \l__stex_modules_group_depth_int
1127 \tl_new:N \l__stex_modules_aftergroup_tl
1128 \cs_new_protected:Nn \stex_do_up_to_module:n {
1129   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1130     #1
1131   }{
1132     #1
1133     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1134     \aftergroup\__stex_modules_aftergroup_do:
1135   }
1136 }
1137 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1138   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1139     \l__stex_modules_aftergroup_tl
1140     \tl_clear:N \l__stex_modules_aftergroup_tl
1141   }{
1142     \l__stex_modules_aftergroup_tl
1143     \aftergroup\__stex_modules_aftergroup_do:
1144   }
1145 }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 29.)

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

1146

(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.

```

1147 \str_new:N \l_stex_modules_ns_str
1148 \str_new:N \l_stex_modules_subpath_str

```

```

1149 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1150   \str_set:Nx \l_tmpa_str { #1 }
1151   \seq_set_eq:NN \l_tmpa_seq #2
1152   % split off file extension
1153   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1154   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1155   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1156   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1157
1158   \bool_set_true:N \l_tmpa_bool
1159   \bool_while_do:Nn \l_tmpa_bool {
1160     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1161     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1162       {source} { \bool_set_false:N \l_tmpa_bool }
1163     }{}{
1164       \seq_if_empty:NT \l_tmpa_seq {
1165         \bool_set_false:N \l_tmpa_bool
1166       }
1167     }
1168   }
1169
1170   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1171   \str_if_empty:NTF \l_stex_modules_subpath_str {
1172     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1173   }{
1174     \str_set:Nx \l_stex_modules_ns_str {
1175       \l_tmpa_str/\l_stex_modules_subpath_str
1176     }
1177   }
1178 }
1179
1180 \cs_new_protected:Nn \stex_modules_current_namespace: {
1181   \str_clear:N \l_stex_modules_subpath_str
1182   \prop_if_exist:NTF \l_stex_current_repository_prop {
1183     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1184     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1185   }{
1186     % split off file extension
1187     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1188     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1189     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1190     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1191     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1192     \str_set:Nx \l_stex_modules_ns_str {
1193       file:\stex_path_to_string:N \l_tmpa_seq
1194     }
1195   }
1196 }

```

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

28.1 The smodule environment

smodule arguments:

```

1197 \keys_define:nn { stex / module } {
1198   title      .tl_set:N      = \smodulename ,
1199   type       .str_set_x:N    = \smoduletype ,
1200   id         .str_set_x:N    = \smoduleid ,
1201   deprecate  .str_set_x:N    = \l_stex_module_deprecate_str ,
1202   ns         .str_set_x:N    = \l_stex_module_ns_str ,
1203   lang       .str_set_x:N    = \l_stex_module_lang_str ,
1204   sig        .str_set_x:N    = \l_stex_module_sig_str ,
1205   creators   .str_set_x:N    = \l_stex_module_creators_str ,
1206   contributors .str_set_x:N  = \l_stex_module_contributors_str ,
1207   meta       .str_set_x:N    = \l_stex_module_meta_str ,
1208   srccite    .str_set_x:N    = \l_stex_module_srccite_str
1209 }
1210
1211 \cs_new_protected:Nn \__stex_modules_args:n {
1212   \str_clear:N \smodulename
1213   \str_clear:N \smoduletype
1214   \str_clear:N \smoduleid
1215   \str_clear:N \l_stex_module_ns_str
1216   \str_clear:N \l_stex_module_deprecate_str
1217   \str_clear:N \l_stex_module_lang_str
1218   \str_clear:N \l_stex_module_sig_str
1219   \str_clear:N \l_stex_module_creators_str
1220   \str_clear:N \l_stex_module_contributors_str
1221   \str_clear:N \l_stex_module_meta_str
1222   \str_clear:N \l_stex_module_srccite_str
1223   \keys_set:nn { stex / module } { #1 }
1224 }
1225
1226 % module parameters here? In the body?
1227

```

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

```

1228 \cs_new_protected:Nn \stex_module_setup:nn {
1229   \str_set:Nx \l_stex_module_name_str { #2 }
1230   \__stex_modules_args:n { #1 }
1231
1232   First, we set up the name and namespace of the module.
1233   Are we in a nested module?
1234
1235   \stex_if_in_module:TF {
1236     % Nested module
1237     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1238       { ns } \l_stex_module_ns_str
1239     \str_set:Nx \l_stex_module_name_str {
1240       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1241       { name } / \l_stex_module_name_str
1242     }
1243   }{
1244     % not nested:
1245     \str_if_empty:NT \l_stex_module_ns_str {
1246       \stex_modules_current_namespace:
1247     }
1248   }
1249 }

```

```

1243 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1244 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1245 / {\l_stex_module_ns_str}
1246 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1247 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1248 \str_set:Nx \l_stex_module_ns_str {
1249 \stex_path_to_string:N \l_tmpa_seq
1250 }
1251 }
1252 }
1253 }

```

Next, we determine the language of the module:

```

1254 \str_if_empty:NT \l_stex_module_lang_str {
1255 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1256 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1257 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1258 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1259 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1260 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1261 inferred~from~file~name}
1262 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1263 }
1264 }
1265
1266 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1267 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1268 \l_tmpa_str {
1269 \ltx@ifpackageloaded{babel}{
1270 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1271 }{}
1272 } {
1273 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1274 }
1275 }}

```

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

```

1276 \str_if_empty:NTF \l_stex_module_sig_str {
1277 \exp_args:Nnx \prop_gset_from_keyval:cn {
1278 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1279 } {
1280 name = \l_stex_module_name_str ,
1281 ns = \l_stex_module_ns_str ,
1282 file = \exp_not:o { \g_stex_currentfile_seq } ,
1283 lang = \l_stex_module_lang_str ,
1284 sig = \l_stex_module_sig_str ,
1285 deprecate = \l_stex_module_deprecate_str ,
1286 meta = \l_stex_module_meta_str
1287 }
1288 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1289 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1290 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1291 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1292 \str_if_empty:NT \l_stex_module_meta_str {
1293   \str_set:Nx \l_stex_module_meta_str {
1294     \c_stex_metatheory_ns_str ? Metatheory
1295   }
1296 }
1297 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1298   \bool_set_true:N \l_stex_in_meta_bool
1299   \exp_args:Nx \stex_add_to_current_module:n {
1300     \bool_set_true:N \l_stex_in_meta_bool
1301     \stex_activate_module:n {\l_stex_module_meta_str}
1302     \bool_set_false:N \l_stex_in_meta_bool
1303   }
1304   \stex_activate_module:n {\l_stex_module_meta_str}
1305   \bool_set_false:N \l_stex_in_meta_bool
1306 }
1307 }{
1308   \str_if_empty:NT \l_stex_module_lang_str {
1309     \msg_error:nnxx{stex}{error/siglanguage}{
1310       \l_stex_module_ns_str?\l_stex_module_name_str
1311     }{\l_stex_module_sig_str}
1312   }
1313
1314   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1315   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1316   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1317   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1318   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1319   \str_set:Nx \l_tmpa_str {
1320     \stex_path_to_string:N \l_tmpa_seq /
1321     \l_tmpa_str . \l_stex_module_sig_str .tex
1322   }
1323   \IfFileExists \l_tmpa_str {
1324     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1325       \str_clear:N \l_stex_current_module_str
1326       \seq_clear:N \l_stex_all_modules_seq
1327       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1328     }
1329   }{
1330     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1331   }
1332   \stex_if_smsmode:F {
1333     \stex_activate_module:n {
1334       \l_stex_module_ns_str ? \l_stex_module_name_str
1335     }
1336   }
1337   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1338 }
1339 \str_if_empty:NF \l_stex_module_deprecate_str {
1340   \msg_warning:nnxx{stex}{warning/deprecated}{
1341     Module~\l_stex_current_module_str
1342   }{
1343     \l_stex_module_deprecate_str
1344   }

```

```

1345 }
1346 \seq_put_right:Nx \l_stex_all_modules_seq {
1347   \l_stex_module_ns_str ? \l_stex_module_name_str
1348 }
1349 }

```

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

smodule The module environment.

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

```

1350 \cs_new_protected:Nn \_stex_modules_begin_module: {
1351   \stex_reactivate_macro:N \STEXexport
1352   \stex_reactivate_macro:N \importmodule
1353   \stex_reactivate_macro:N \symdecl
1354   \stex_reactivate_macro:N \notation
1355   \stex_reactivate_macro:N \symdef
1356
1357   \stex_debug:nn{modules}{
1358     New~module:\\
1359     Namespace:~\l_stex_module_ns_str\\
1360     Name:~\l_stex_module_name_str\\
1361     Language:~\l_stex_module_lang_str\\
1362     Signature:~\l_stex_module_sig_str\\
1363     Metatheory:~\l_stex_module_meta_str\\
1364     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1365   }
1366
1367   \stex_if_smsmode:F{
1368     \begin{stex_annotate_env} {theory} {
1369       \l_stex_module_ns_str ? \l_stex_module_name_str
1370     }
1371
1372     \stex_annotate_invisible:nnn{header}{} {
1373       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1374       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1375       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1376         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1377       }
1378       \str_if_empty:NF \smoduletype {
1379         \stex_annotate:nnn{type}{\smoduletype}{}
1380       }
1381     }
1382   }
1383   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1384   % TODO: Inherit metatheory for nested modules?
1385 }
1386 \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}`

```

1387 \cs_new_protected:Nn \_stex_modules_end_module: {
1388   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1389 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1390 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1391 \NewDocumentEnvironment { smodule } { 0{} m } {
1392   \stex_module_setup:nn{#1}{#2}
1393   \par
1394   \stex_if_smsmode:F{
1395     \tl_clear:N \l_tmpa_tl
1396     \clist_map_inline:Nn \smoduletype {
1397       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1398         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1399       }
1400     }
1401     \tl_if_empty:NTF \l_tmpa_tl {
1402       \__stex_modules_smodule_start:
1403     }{
1404       \l_tmpa_tl
1405     }
1406   }
1407   \__stex_modules_begin_module:
1408   \str_if_empty:NF \smoduleid {
1409     \stex_ref_new_doc_target:n \smoduleid
1410   }
1411   \stex_smsmode_do:
1412 } {
1413   \__stex_modules_end_module:
1414   \stex_if_smsmode:F {
1415     \end{stex_annotate_env}
1416     \clist_set:Nn \l_tmpa_clist \smoduletype
1417     \tl_clear:N \l_tmpa_tl
1418     \clist_map_inline:Nn \l_tmpa_clist {
1419       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1420         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1421       }
1422     }
1423     \tl_if_empty:NTF \l_tmpa_tl {
1424       \__stex_modules_smodule_end:
1425     }{
1426       \l_tmpa_tl
1427     }
1428   }
1429 }

```

`\stexpatchmodule`

```

1430 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1431 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1432
1433 \newcommand\stexpatchmodule[3] [] {
1434   \str_set:Nx \l_tmpa_str{ #1 }
1435   \str_if_empty:NTF \l_tmpa_str {
1436     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1437     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1438   }{

```

```

1439     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1440     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1441   }
1442 }

```

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

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1443 \NewDocumentCommand \STEXModule { m } {
1444   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1445   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1446   \tl_set:Nn \l_tmpa_tl {
1447     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1448   }
1449   \seq_map_inline:Nn \l_stex_all_modules_seq {
1450     \str_set:Nn \l_tmpb_str { ##1 }
1451     \str_if_eq:eeT { \l_tmpa_str } {
1452       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1453     } {
1454       \seq_map_break:n {
1455         \tl_set:Nn \l_tmpa_tl {
1456           \stex_invoke_module:n { ##1 }
1457         }
1458       }
1459     }
1460   }
1461   \l_tmpa_tl
1462 }
1463
1464 \cs_new_protected:Nn \stex_invoke_module:n {
1465   \stex_debug:nn{modules}{Invoking~module~#1}
1466   \peek_charcode_remove:NTF ! {
1467     \__stex_modules_invoke_uri:nN { #1 }
1468   } {
1469     \peek_charcode_remove:NTF ? {
1470       \__stex_modules_invoke_symbol:nn { #1 }
1471     } {
1472       \msg_error:nnx{stex}{error/syntax}{
1473         ?~or~!~expected~after~
1474         \c_backslash_str STEXModule{#1}
1475       }
1476     }
1477   }
1478 }
1479
1480 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1481   \str_set:Nn #2 { #1 }
1482 }
1483
1484 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1485   \stex_invoke_symbol:n{#1?#2}

```



```
1486 }
```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 30.)

`\stex_activate_module:n`

```
1487 \bool_new:N \l_stex_in_meta_bool
1488 \bool_set_false:N \l_stex_in_meta_bool
1489 \cs_new_protected:Nn \stex_activate_module:n {
1490   \stex_debug:nn{modules}{Activating~module~#1}
1491   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1492     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1493   }
1494   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1495     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1496     \use:c{ c_stex_module_#1_code }
1497   }
1498 }
```

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

```
1499 \</package>
```

Chapter 29

STEX -Module Inheritance Implementation

```
1500 <*package>
1501
1502 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1503
```

29.1 SMS Mode

```
1504 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1505 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1506 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1507 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1508
1509 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1510   \makeatletter
1511   \makeatother
1512   \ExplSyntaxOn
1513   \ExplSyntaxOff
1514   \rustexBREAK
1515 }
1516
1517 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1518   \symdef
1519   \importmodule
1520   \notation
1521   \symdecl
1522   \STEXexport
1523   \inlineass
1524   \inlinedef
1525   \inlineex
1526   \endinput
1527   \setnotation
```

```

1528 \copynotation
1529 }
1530
1531 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1532   \tl_to_str:n {
1533     smodule,
1534     copymodule,
1535     interpretmodule,
1536     sdefinition,
1537     sexample,
1538     sassertion,
1539     sparagraph
1540   }
1541 }

```

(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 32.)

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```

1542 \bool_new:N \g__stex_smsmode_bool
1543 \bool_set_false:N \g__stex_smsmode_bool
1544 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1545   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1546 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1547 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1548   \vbox_set:Nn \l_tmpa_box {
1549     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1550     \bool_gset_true:N \g__stex_smsmode_bool
1551     #2
1552     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1553   }
1554   \box_clear:N \l_tmpa_box
1555 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1556 \quark_new:N \q__stex_smsmode_break
1557
1558 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1559   \stex_filestack_push:n{#1}
1560   \_stex_smsmode_in_smsmode:nn{#1} {
1561     #2
1562     \everyeof{\q__stex_smsmode_break\noexpand}
1563     \expandafter\expandafter\expandafter
1564     \stex_smsmode_do:
1565     \csname @ @ input\endcsname "#1"\relax
1566   }
1567   \stex_filestack_pop:
1568 }

```

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

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

```

1569 \cs_new_protected:Npn \stex_smsmode_do: {
1570   \stex_if_smsmode:T {
1571     \__stex_smsmode_do:w
1572   }
1573 }
1574 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1575   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1576     \expandafter\if\expandafter\relax\noexpand#1
1577     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1578   }{
1579     \else\expandafter\__stex_smsmode_do:w\fi
1580   }{
1581     \__stex_smsmode_do:w % #1
1582   }
1583 }
1584 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1585   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1586     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1587       #1\__stex_smsmode_do:w
1588     }{
1589       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1590         #1
1591       }{
1592         \cs_if_eq:NNTF \begin #1 {
1593           \__stex_smsmode_check_begin:n
1594         }{
1595           \cs_if_eq:NNTF \end #1 {
1596             \__stex_smsmode_check_end:n
1597           }{
1598             \__stex_smsmode_do:w
1599           }
1600         }
1601       }
1602     }
1603   }
1604 }
1605 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1606   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1607     \begin{#1}
1608   }{
1609     \__stex_smsmode_do:w
1610   }
1611 }
1612 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1613   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1614     \end{#1}\__stex_smsmode_do:w
1615   }{
1616     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1617   }
1618 }

```

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

29.2 Inheritance

```

1619 <@@=stex_importmodule>

\stex_import_module_uri:nn

1620 \cs_new_protected:Nn \stex_import_module_uri:nn {
1621   \str_set:Nx \l_stex_import_archive_str { #1 }
1622   \str_set:Nn \l_stex_import_path_str { #2 }
1623
1624   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1625   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1626   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1627
1628   \stex_modules_current_namespace:
1629   \bool_lazy_all:nTF {
1630     {\str_if_empty_p:N \l_stex_import_archive_str}
1631     {\str_if_empty_p:N \l_stex_import_path_str}
1632     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1633   }{
1634     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1635     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1636   }{
1637     \str_if_empty:NT \l_stex_import_archive_str {
1638       \prop_if_exist:NT \l_stex_current_repository_prop {
1639         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1640       }
1641     }
1642     \str_if_empty:NTF \l_stex_import_archive_str {
1643       \str_if_empty:NF \l_stex_import_path_str {
1644         \str_set:Nx \l_stex_import_ns_str {
1645           \l_stex_module_ns_str / \l_stex_import_path_str
1646         }
1647       }
1648     }{
1649       \stex_require_repository:n \l_stex_import_archive_str
1650       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1651       \l_stex_import_ns_str
1652       \str_if_empty:NF \l_stex_import_path_str {
1653         \str_set:Nx \l_stex_import_ns_str {
1654           \l_stex_import_ns_str / \l_stex_import_path_str
1655         }
1656       }
1657     }
1658   }
1659 }
```

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

<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>	<code>\str_new:N \l_stex_import_name_str</code>
<code>\l_stex_import_path_str</code>	<code>\str_new:N \l_stex_import_archive_str</code>
<code>\l_stex_import_ns_str</code>	<code>\str_new:N \l_stex_import_path_str</code>

```
1663 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 34.)

```
\stex_import_require_module:nnnn
    {\ns} {\{archive-ID\}} {\{path\}} {\{name\}}
1664 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1665   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1666
1667     % archive
1668     \str_set:Nx \l_tmpa_str { #2 }
1669     \str_if_empty:NTF \l_tmpa_str {
1670       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1671     } {
1672       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1673       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1674       \seq_put_right:Nn \l_tmpa_seq { source }
1675     }
1676
1677     % path
1678     \str_set:Nx \l_tmpb_str { #3 }
1679     \str_if_empty:NTF \l_tmpb_str {
1680       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1681
1682       \ltx@ifpackageloaded{babel} {
1683         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1684           { \language } \l_tmpb_str {
1685           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1686         }
1687       } {
1688         \str_clear:N \l_tmpb_str
1689       }
1690
1691       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1692       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1693         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1694       }{
1695         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1696         \IfFileExists{ \l_tmpa_str.tex }{
1697           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1698         }{
1699           % try english as default
1700           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1701           \IfFileExists{ \l_tmpa_str.en.tex }{
1702             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1703           }{
1704             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1705           }
1706         }
1707       }
1708
1709     } {
1710       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1711       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1712
```

```

1713 \ltx@ifpackageloaded{babel} {
1714   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1715   { \language } \l_tmpb_str {
1716     \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1717   }
1718 } {
1719   \str_clear:N \l_tmpb_str
1720 }
1721
1722 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1723
1724 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1725 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1726   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1727 }{
1728   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1729   \IfFileExists{ \l_tmpa_str/#4.tex }{
1730     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1731   }{
1732     % try english as default
1733     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1734     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1735       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1736     }{
1737       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1738       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1739         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1740       }{
1741         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1742         \IfFileExists{ \l_tmpa_str.tex }{
1743           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1744         }{
1745           % try english as default
1746           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1747           \IfFileExists{ \l_tmpa_str.en.tex }{
1748             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1749           }{
1750             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1751           }
1752         }
1753       }
1754     }
1755   }
1756 }
1757 }
1758
1759 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1760   \seq_clear:N \l_stex_all_modules_seq
1761   \str_clear:N \l_stex_current_module_str
1762   \str_set:Nx \l_tmpb_str { #2 }
1763   \str_if_empty:NF \l_tmpb_str {
1764     \stex_set_current_repository:n { #2 }
1765   }
1766   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1767     }
1768
1769     \stex_if_module_exists:nF { #1 ? #4 } {
1770       \msg_error:nnx{stex}{error/unknownmodule}{
1771         #1?#4~(in~file~\g__stex_importmodule_file_str)
1772       }
1773     }
1774   }
1775   \stex_activate_module:n { #1 ? #4 }
1776 }

```

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

`\importmodule`

```

1777 \NewDocumentCommand \importmodule { 0{} m } {
1778   \stex_import_module_uri:nn { #1 } { #2 }
1779   \stex_debug:nn{modules}{Importing~module:~
1780     \l_stex_import_ns_str ? \l_stex_import_name_str
1781   }
1782   \stex_if_smsmode:F {
1783     \stex_import_require_module:nnnn
1784     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1785     { \l_stex_import_path_str } { \l_stex_import_name_str }
1786     \stex_annotate_invisible:nnn
1787     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1788   }
1789   \exp_args:Nx \stex_add_to_current_module:n {
1790     \stex_import_require_module:nnnn
1791     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1792     { \l_stex_import_path_str } { \l_stex_import_name_str }
1793   }
1794   \exp_args:Nx \stex_add_import_to_current_module:n {
1795     \l_stex_import_ns_str ? \l_stex_import_name_str
1796   }
1797   \stex_smsmode_do:
1798   \ignorespacesandpars
1799 }
1800 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

1801 \NewDocumentCommand \usemodule { 0{} m } {
1802   \stex_if_smsmode:F {
1803     \stex_import_module_uri:nn { #1 } { #2 }
1804     \stex_import_require_module:nnnn
1805     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1806     { \l_stex_import_path_str } { \l_stex_import_name_str }
1807     \stex_annotate_invisible:nnn
1808     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1809   }
1810   \stex_smsmode_do:
1811   \ignorespacesandpars
1812 }

```


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

1813 `\endpackage`

Chapter 30

STEX -Symbols Implementation

```
1814 <*package>
1815
1816 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1817
      Warnings and error messages
1818 \msg_new:nnn{stex}{error/wrongargs}{
1819   args~value~in~symbol~declaration~for~#1~
1820   needs~to~be~i,~a,~b~or~B,~but~#2~given
1821 }
1822 \msg_new:nnn{stex}{error/unknownsymbol}{
1823   No~symbol~#1~found!
1824 }
1825 \msg_new:nnn{stex}{error/seqlength}{
1826   Expected~#1~arguments;~got~#2!
1827 }
```

30.1 Symbol Declarations

```
1828 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols
1829 \cs_new_protected:Nn \stex_all_symbols:n {
1830   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1831   \seq_map_inline:Nn \l_stex_all_modules_seq {
1832     \seq_map_inline:cn{c_stex_module_##1_constants}{
1833       \__stex_symdecl_all_symbols_cs{##1?####1}
1834     }
1835   }
1836 }

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

\STEXsymbol
1837 \NewDocumentCommand \STEXsymbol { m } {
1838   \stex_get_symbol:n { #1 }
```

```

1839 \exp_args:No
1840 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1841 }

```

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

`symdecl` arguments:

```

1842 \keys_define:nn { stex / symdecl } {
1843   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1844   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1845   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1846   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1847   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1848   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1849   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1850   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1851   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1852   assoc     .choices:nn =
1853     {bin,binl,binr,pre,conj,pwconj}
1854     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1855 }
1856
1857 \bool_new:N \l_stex_symdecl_make_macro_bool
1858
1859 \cs_new_protected:Nn \__stex_symdecl_args:n {
1860   \str_clear:N \l_stex_symdecl_name_str
1861   \str_clear:N \l_stex_symdecl_args_str
1862   \str_clear:N \l_stex_symdecl_deprecate_str
1863   \str_clear:N \l_stex_symdecl_astype_str
1864   \bool_set_false:N \l_stex_symdecl_local_bool
1865   \tl_clear:N \l_stex_symdecl_type_tl
1866   \tl_clear:N \l_stex_symdecl_definiens_tl
1867
1868   \keys_set:nn { stex / symdecl } { #1 }
1869 }

```

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

```

1870
1871 \NewDocumentCommand \symdecl { s m O{} } {
1872   \__stex_symdecl_args:n { #3 }
1873   \IfBooleanTF #1 {
1874     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1875   } {
1876     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1877   }
1878   \stex_symdecl_do:n { #2 }
1879   \stex_smsmode_do:
1880 }
1881
1882 \cs_new_protected:Nn \stex_symdecl_do:nn {
1883   \__stex_symdecl_args:n{#1}
1884   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1885   \stex_symdecl_do:n{#2}
1886 }

```

```

1887
1888 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1889 \cs_new_protected:Nn \stex_symdecl_do:n {
1890   \stex_if_in_module:F {
1891     % TODO throw error? some default namespace?
1892   }
1893
1894   \str_if_empty:NT \l_stex_symdecl_name_str {
1895     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1896   }
1897
1898   \prop_if_exist:cT { l_stex_symdecl_
1899     \l_stex_current_module_str ?
1900     \l_stex_symdecl_name_str
1901   }_prop
1902   {
1903     % TODO throw error (beware of circular dependencies)
1904   }
1905
1906   \prop_clear:N \l_tmpa_prop
1907   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1908   \seq_clear:N \l_tmpa_seq
1909   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1910   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1911
1912   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1913     \str_if_empty:NF \l_stex_module_deprecate_str {
1914       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1915     }
1916   }
1917   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1918
1919   \exp_args:No \stex_add_constant_to_current_module:n {
1920     \l_stex_symdecl_name_str
1921   }
1922
1923   % arity/args
1924   \int_zero:N \l_tmpb_int
1925
1926   \bool_set_true:N \l_tmpa_bool
1927   \str_map_inline:Nn \l_stex_symdecl_args_str {
1928     \token_case_meaning:NnF ##1 {
1929       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1930       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1931       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1932       {\tl_to_str:n a} {
1933         \bool_set_false:N \l_tmpa_bool
1934         \int_incr:N \l_tmpb_int
1935       }
1936       {\tl_to_str:n B} {

```

```

1937     \bool_set_false:N \l_tmpa_bool
1938     \int_incr:N \l_tmpb_int
1939   }
1940 }{
1941   \msg_error:nnxx{stex}{error/wrongargs}{
1942     \l_stex_current_module_str ?
1943     \l_stex_symdecl_name_str
1944   }{##1}
1945 }
1946 }
1947 \bool_if:NTF \l_tmpa_bool {
1948   % possibly numeric
1949   \str_if_empty:NTF \l_stex_symdecl_args_str {
1950     \prop_put:Nnn \l_tmpa_prop { args } {}
1951     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1952   }{
1953     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1954     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1955     \str_clear:N \l_tmpa_str
1956     \int_step_inline:nn \l_tmpa_int {
1957       \str_put_right:Nn \l_tmpa_str i
1958     }
1959     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1960   }
1961 } {
1962   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1963   \prop_put:Nnx \l_tmpa_prop { arity }
1964     { \str_count:N \l_stex_symdecl_args_str }
1965 }
1966 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1967
1968
1969 % semantic macro
1970
1971 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1972   \exp_args:Nx \stex_do_up_to_module:n {
1973     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1974       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1975     }}
1976   }
1977
1978   \bool_if:NF \l_stex_symdecl_local_bool {
1979     \exp_args:Nx \stex_add_to_current_module:n {
1980       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1981         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1982       } }
1983     }
1984   }
1985 }
1986
1987 \stex_debug:nn{symbols}{New~symbol:~
1988   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1989   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1990   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J

```

```

1991     Definien:~\exp_not:o {\l_stex_symdecl_definiens_tl}
1992 }
1993
1994 % circular dependencies require this:
1995
1996 \prop_if_exist:cF {
1997     \l_stex_symdecl_
1998     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1999     _prop
2000 } {
2001     \exp_args:Nx \stex_do_up_to_module:n {
2002         \prop_set_from_keyval:cn {
2003             \l_stex_symdecl_
2004             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2005             _prop
2006         } {\prop_to_keyval:N \l_tmpa_prop}
2007     }
2008 }
2009
2010 \seq_clear:c {
2011     \l_stex_symdecl_
2012     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013     _notations
2014 }
2015
2016 \bool_if:NF \l_stex_symdecl_local_bool {
2017     \exp_args:Nx
2018     \stex_add_to_current_module:n {
2019         \seq_clear:c {
2020             \l_stex_symdecl_
2021             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2022             _notations
2023         }
2024         \prop_set_from_keyval:cn {
2025             \l_stex_symdecl_
2026             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027             _prop
2028         } {
2029             name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2030             module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2031             type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2032             args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2033             arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2034             assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2035         }
2036     }
2037 }
2038
2039 \stex_if_smsmode:F {
2040 %     \exp_args:Nx \stex_do_up_to_module:n {
2041 %         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2042 %             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2043 %         }
2044 %     }

```

```

2045 \stex_if_do_html:T {
2046   \stex_annotate_invisible:nnn {symdecl} {
2047     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2048   } {
2049     \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2050     \stex_annotate_invisible:nnn{args}{}}{
2051       \prop_item:Nn \l_tmpa_prop { args }
2052     }
2053     \stex_annotate_invisible:nnn{macroname}{#1}{}
2054     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2055       \stex_annotate_invisible:nnn{definiens}{}
2056       {\l_stex_symdecl_definiens_tl$}
2057     }
2058     \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2059       \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2060     }
2061   }
2062 }
2063 }
2064 }

```

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

`\stex_get_symbol:n`

```

2065 \str_new:N \l_stex_get_symbol_uri_str
2066
2067 \cs_new_protected:Nn \stex_get_symbol:n {
2068   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2069     \tl_set:Nn \l_tmpa_tl { #1 }
2070     \__stex_symdecl_get_symbol_from_cs:
2071   }{
2072     % argument is a string
2073     % is it a command name?
2074     \cs_if_exist:cTF { #1 }{
2075       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2076       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2077       \str_if_empty:NTF \l_tmpa_str {
2078         \exp_args:Nx \cs_if_eq:NNTF {
2079           \tl_head:N \l_tmpa_tl
2080         } \stex_invoke_symbol:n {
2081           \__stex_symdecl_get_symbol_from_cs:
2082         }{
2083           \__stex_symdecl_get_symbol_from_string:n { #1 }
2084         }
2085       } {
2086         \__stex_symdecl_get_symbol_from_string:n { #1 }
2087       }
2088     }{
2089       % argument is not a command name
2090       \__stex_symdecl_get_symbol_from_string:n { #1 }
2091       % \l_stex_all_symbols_seq
2092     }
2093   }
2094   \str_if_eq:eeF {

```

```

2095 \prop_item:cn {
2096   l_stex_symdecl~\l_stex_get_symbol_uri_str _prop
2097 }{ deprecate }
2098 }{}{
2099 \msg_warning:nnxx{stex}{warning/deprecated}{
2100   Symbol~\l_stex_get_symbol_uri_str
2101 }{
2102   \prop_item:cn {l_stex_symdecl~\l_stex_get_symbol_uri_str _prop}{ deprecate }
2103 }
2104 }
2105 }
2106
2107 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2108   \tl_set:Nn \l_tmpa_tl {
2109     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2110   }
2111   \str_set:Nn \l_tmpa_str { #1 }
2112   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2113
2114   \stex_all_symbols:n {
2115     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2116       \seq_map_break:n{\seq_map_break:n{
2117         \tl_set:Nn \l_tmpa_tl {
2118           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2119         }
2120       }}
2121     }
2122   }
2123
2124   \l_tmpa_tl
2125 }
2126
2127 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2128   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2129   { \tl_tail:N \l_tmpa_tl }
2130   \tl_if_single:NTF \l_tmpa_tl {
2131     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2132       \exp_after:wN \str_set:Nn \exp_after:wN
2133       \l_stex_get_symbol_uri_str \l_tmpa_tl
2134     }{
2135       % TODO
2136       % tail is not a single group
2137     }
2138   }{
2139     % TODO
2140     % tail is not a single group
2141   }
2142 }

```

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

30.2 Notations

```

2143 <@@=stex_notation>

```



```

notation arguments:
2144 \keys_define:nn { stex / notation } {
2145   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2146   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2147   prec .str_set_x:N = \l__stex_notation_prec_str ,
2148   op .tl_set:N = \l__stex_notation_op_tl ,
2149   primary .bool_set:N = \l__stex_notation_primary_bool ,
2150   primary .default:n = {true} ,
2151   unknown .code:n = \str_set:Nx
2152     \l__stex_notation_variant_str \l_keys_key_str
2153 }
2154
2155 \cs_new_protected:Nn \_stex_notation_args:n {
2156   \str_clear:N \l__stex_notation_lang_str
2157   \str_clear:N \l__stex_notation_variant_str
2158   \str_clear:N \l__stex_notation_prec_str
2159   \tl_clear:N \l__stex_notation_op_tl
2160   \bool_set_false:N \l__stex_notation_primary_bool
2161
2162   \keys_set:nn { stex / notation } { #1 }
2163 }

```

\notation

```

2164 \NewDocumentCommand \notation { s m O{}} {
2165   \_stex_notation_args:n { #3 }
2166   \tl_clear:N \l_stex_symdecl_definiens_tl
2167   \stex_get_symbol:n { #2 }
2168   \tl_set:Nn \l_stex_notation_after_do_tl {
2169     \__stex_notation_final:
2170     \IfBooleanTF#1{
2171       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2172     }{}
2173     \stex_smsmode_do:
2174   }
2175   \stex_notation_do:nnnnn
2176   { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop} { args } }
2177   { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
2178   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2179   { \l__stex_notation_prec_str }
2180 }
2181 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nnnnn

```

2182 \seq_new:N \l__stex_notation_precedences_seq
2183 \tl_new:N \l__stex_notation_opprec_tl
2184 \int_new:N \l__stex_notation_currarg_int
2185 \tl_new:N \stex_symbol_after_invokation_tl
2186
2187 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2188   \let\l_stex_current_symbol_str\relax
2189   \seq_clear:N \l__stex_notation_precedences_seq
2190   \tl_clear:N \l__stex_notation_opprec_tl
2191   \str_set:Nx \l__stex_notation_args_str { #1 }

```

```

2192 \str_set:Nx \l__stex_notation_arity_str { #2 }
2193 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2194 \str_set:Nx \l__stex_notation_prec_str { #4 }
2195
2196 % precedences
2197 \str_if_empty:NTF \l__stex_notation_prec_str {
2198   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2199     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2200   }{
2201     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2202   }
2203 } {
2204   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2205     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2206     \int_step_inline:nn { \l__stex_notation_arity_str } {
2207       \exp_args:NNo
2208       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2209     }
2210   }{
2211     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2212     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2213       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2214       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2215         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2216         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2217         \seq_map_inline:Nn \l_tmpa_seq {
2218           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2219         }
2220       }
2221     }{
2222       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2223         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2224       }{
2225         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2226       }
2227     }
2228   }
2229 }
2230
2231 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2232 \int_step_inline:nn { \l__stex_notation_arity_str } {
2233   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2234     \exp_args:NNo
2235     \seq_put_right:No \l__stex_notation_precedences_seq {
2236       \l__stex_notation_opprec_tl
2237     }
2238   }
2239 }
2240 \tl_clear:N \l_stex_notation_dummyargs_tl
2241
2242 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2243   \exp_args:NNe
2244   \cs_set:Npn \l_stex_notation_macrocode_cs {
2245     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }

```

```

2246     { \l__stex_notation_suffix_str }
2247     { \l__stex_notation_opprec_tl }
2248     { \exp_not:n { #5 } }
2249   }
2250   \l_stex_notation_after_do_tl
2251 }{
2252   \str_if_in:NnTF \l__stex_notation_args_str b {
2253     \exp_args:Nne \use:nn
2254     {
2255       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2256       \cs_set:Npn \l__stex_notation_arity_str } { {
2257         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2258         { \l__stex_notation_suffix_str }
2259         { \l__stex_notation_opprec_tl }
2260         { \exp_not:n { #5 } }
2261       }}
2262   }{
2263     \str_if_in:NnTF \l__stex_notation_args_str B {
2264       \exp_args:Nne \use:nn
2265       {
2266         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2267         \cs_set:Npn \l__stex_notation_arity_str } { {
2268           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2269           { \l__stex_notation_suffix_str }
2270           { \l__stex_notation_opprec_tl }
2271           { \exp_not:n { #5 } }
2272         } }
2273   }{
2274     \exp_args:Nne \use:nn
2275     {
2276       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2277       \cs_set:Npn \l__stex_notation_arity_str } { {
2278         \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2279         { \l__stex_notation_suffix_str }
2280         { \l__stex_notation_opprec_tl }
2281         { \exp_not:n { #5 } }
2282       } }
2283   }
2284 }
2285
2286 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2287 \int_zero:N \l__stex_notation_currarg_int
2288 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2289 \l__stex_notation_arguments:
2290 }
2291 }

```

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

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

```

2292 \cs_new_protected:Nn \__stex_notation_arguments: {
2293   \int_incr:N \l__stex_notation_currarg_int
2294   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2295     \l_stex_notation_after_do_tl

```

```

2296 }{
2297   \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2298   \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2299   \str_if_eq:VnTF \l_tmpa_str a {
2300     \__stex_notation_argument_assoc:n
2301   }{
2302     \str_if_eq:VnTF \l_tmpa_str B {
2303       \__stex_notation_argument_assoc:n
2304     }{
2305       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2306       \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2307         { \stex_term_math_arg:nnn
2308           { \int_use:N \l__stex_notation_currarg_int }
2309           { \l_tmpa_str }
2310           { ####\int_use:N \l__stex_notation_currarg_int }
2311         }
2312       }
2313       \__stex_notation_arguments:
2314     }
2315   }
2316 }
2317 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2318 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2319
2320   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2321     {\l__stex_notation_arity_str}{
2322       #1
2323     }
2324   \int_zero:N \l_tmpa_int
2325   \tl_clear:N \l_tmpa_tl
2326   \str_map_inline:Nn \l__stex_notation_args_str {
2327     \int_incr:N \l_tmpa_int
2328     \tl_put_right:Nx \l_tmpa_tl {
2329       \str_if_eq:nnTF {##1}{a}{ {} } {
2330         \str_if_eq:nnTF {##1}{B}{ {} } {
2331           {\stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2332         }
2333       }
2334     }
2335   }
2336   \exp_after:wN\exp_after:wN\exp_after:wN \def
2337   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2338   \exp_after:wN\exp_after:wN\exp_after:wN ##
2339   \exp_after:wN\exp_after:wN\exp_after:wN 1
2340   \exp_after:wN\exp_after:wN\exp_after:wN ##
2341   \exp_after:wN\exp_after:wN\exp_after:wN 2
2342   \exp_after:wN\exp_after:wN\exp_after:wN {
2343     \exp_after:wN \exp_after:wN \exp_after:wN
2344     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2345       \exp_after:wN \l_tmpa_cs \l_tmpa_tl

```

```

2346     }
2347 }
2348
2349 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2350 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2351   \stex_term_math_assoc_arg:nnnn
2352   { \int_use:N \l__stex_notation_currarg_int }
2353   { \l_tmpa_str }
2354   { ####\int_use:N \l__stex_notation_currarg_int }
2355   { \l_tmpa_cs {####1} {####2} }
2356 } }
2357 \__stex_notation_arguments:
2358 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2359 \cs_new_protected:Nn \__stex_notation_final: {
2360   \exp_args:Nne \use:nn
2361   {
2362     \cs_generate_from_arg_count:cNnn {
2363       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2364       \l__stex_notation_suffix_str
2365       _cs
2366     }
2367     \cs_set:Npn \l__stex_notation_arity_str { { {
2368       \exp_after:wN \exp_after:wN \exp_after:wN
2369       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2370       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2371     } } }
2372
2373     \tl_if_empty:NF \l__stex_notation_op_tl {
2374       \cs_set:cpx {
2375         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2376         \l__stex_notation_suffix_str
2377         _cs
2378       } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2379     }
2380
2381     \exp_args:Ne
2382     \stex_add_to_current_module:n {
2383       \cs_generate_from_arg_count:cNnn {
2384         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2385         \l__stex_notation_suffix_str
2386         _cs
2387       } \cs_set:Npn { \l__stex_notation_arity_str } {
2388         \exp_after:wN \exp_after:wN \exp_after:wN
2389         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2390         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2391       }
2392       \tl_if_empty:NF \l__stex_notation_op_tl {
2393         \cs_set:cpn {
2394           stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2395           \l__stex_notation_suffix_str

```

```

2396         _cs
2397     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2398 }
2399 }
2400 %\exp_args:Nx
2401 % \stex_do_up_to_module:n {
2402     \seq_put_right:cx {
2403         \l_stex_symdecl_ \l_stex_get_symbol_uri_str
2404         _notations
2405     } {
2406         \l__stex_notation_suffix_str
2407     }
2408 % }
2409
2410 \stex_debug:nn{symbols}{
2411     Notation~\l__stex_notation_suffix_str
2412     ~for~\l_stex_get_symbol_uri_str^^J
2413     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2414     Argument~precedences:~
2415     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2416     Notation: \cs_meaning:c {
2417         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2418         \l__stex_notation_suffix_str
2419         _cs
2420     }
2421 }
2422
2423 \exp_args:Ne
2424 \stex_add_to_current_module:n {
2425     \seq_put_right:cn {
2426         \l_stex_symdecl_ \l_stex_get_symbol_uri_str
2427         _notations
2428     } { \l__stex_notation_suffix_str }
2429 }
2430
2431 \stex_if_smsmode:F {
2432
2433     % HTML annotations
2434     \stex_if_do_html:T {
2435         \stex_annotate_invisible:nnn { notation }
2436         { \l_stex_get_symbol_uri_str } {
2437             \stex_annotate_invisible:nnn { notationfragment }
2438             { \l__stex_notation_suffix_str }{}
2439             \stex_annotate_invisible:nnn { precedence }
2440             { \l__stex_notation_prec_str }{}
2441
2442             \int_zero:N \l_tmpa_int
2443             \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2444             \tl_clear:N \l_tmpa_tl
2445             \int_step_inline:nn { \l__stex_notation_arity_str }{
2446                 \int_incr:N \l_tmpa_int
2447                 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2448                 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2449                 \str_if_eq:VnTF \l_tmpb_str a {

```

```

2450         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2451             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2452             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2453         } }
2454     }{
2455         \str_if_eq:VnTF \l_tmpb_str B {
2456             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2457                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2458                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2459             } }
2460         }{
2461             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2462                 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2463             } }
2464         }
2465     }
2466 }
2467 \stex_annotate_invisible:nnn { notationcomp }{}{
2468     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2469     $ \exp_args:Nno \use:nn { \use:c {
2470         stex_notation_ \l_stex_current_symbol_str
2471         \c_hash_str \l__stex_notation_suffix_str _cs
2472     } } { \l_tmpa_tl } $
2473 }
2474 }
2475 }
2476 }
2477 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

2478 \keys_define:nn { stex / setnotation } {
2479     lang .tl_set_x:N = \l__stex_notation_lang_str ,
2480     variant .tl_set_x:N = \l__stex_notation_variant_str ,
2481     unknown .code:n = \str_set:Nx
2482         \l__stex_notation_variant_str \l_keys_key_str
2483 }
2484
2485 \cs_new_protected:Nn \stex_setnotation_args:n {
2486     \str_clear:N \l__stex_notation_lang_str
2487     \str_clear:N \l__stex_notation_variant_str
2488     \keys_set:nn { stex / setnotation } { #1 }
2489 }
2490
2491 \cs_new_protected:Nn \stex_setnotation:n {
2492     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2493     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2494         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2495         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2496         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2497         { \c_hash_str }
2498         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2499         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2500 \exp_args:Nx \stex_add_to_current_module:n {
2501   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2502   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2503   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2504   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2505   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2506   { \c_hash_str }
2507 }
2508 \stex_debug:nn {notations}{
2509   Setting~default~notation~
2510   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2511   #1 \\\
2512   \expandafter\meaning\csname
2513   l_stex_symdecl_#1 _notations\endcsname
2514 }
2515 }{
2516   % todo throw error
2517 }
2518 }
2519
2520 \NewDocumentCommand \setnotation {m m} {
2521   \stex_get_symbol:n { #1 }
2522   \_stex_setnotation_args:n { #2 }
2523   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2524   \stex_smsmode_do:
2525 }
2526
2527 \cs_new_protected:Nn \stex_copy_notations:nn {
2528   \stex_debug:nn {notations}{
2529     Copying~notations~from~#2~to~#1\\
2530     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2531   }
2532   \tl_clear:N \l_tmpa_tl
2533   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2534     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2535   }
2536   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2537     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2538     \edef \l_tmpa_tl {
2539       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2540       \exp_after:wN\exp_after:wN\exp_after:wN {
2541         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2542       }
2543     }
2544     \exp_args:Nx
2545     \stex_do_up_to_module:n {
2546       \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2547       \cs_generate_from_arg_count:cNnn {
2548         stex_notation_ #1 \c_hash_str ##1 _cs
2549       } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2550         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2551       }
2552     }
2553   }

```



```

2554 }
2555
2556 \NewDocumentCommand \copynotation {m m} {
2557   \stex_get_symbol:n { #1 }
2558   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2559   \stex_get_symbol:n { #2 }
2560   \exp_args:Noo
2561   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2562   \exp_args:Nx \stex_add_import_to_current_module:n{
2563     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2564   }
2565   \stex_smsmode_do:
2566 }
2567

```

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

\symdef

```

2568 \keys_define:nn { stex / symdef } {
2569   name .str_set_x:N = \l_stex_symdecl_name_str ,
2570   local .bool_set:N = \l_stex_symdecl_local_bool ,
2571   args .str_set_x:N = \l_stex_symdecl_args_str ,
2572   type .tl_set:N = \l_stex_symdecl_type_tl ,
2573   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2574   op .tl_set:N = \l__stex_notation_op_tl ,
2575   lang .str_set_x:N = \l__stex_notation_lang_str ,
2576   variant .str_set_x:N = \l__stex_notation_variant_str ,
2577   prec .str_set_x:N = \l__stex_notation_prec_str ,
2578   assoc .choices:nn =
2579     {bin,binl,binr,pre,conj,pwconj}
2580     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2581   unknown .code:n = \str_set:Nx
2582     \l__stex_notation_variant_str \l_keys_key_str
2583 }
2584
2585 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2586   \str_clear:N \l_stex_symdecl_name_str
2587   \str_clear:N \l_stex_symdecl_args_str
2588   \str_clear:N \l_stex_symdecl_assoctype_str
2589   \bool_set_false:N \l_stex_symdecl_local_bool
2590   \tl_clear:N \l_stex_symdecl_type_tl
2591   \tl_clear:N \l_stex_symdecl_definiens_tl
2592   \str_clear:N \l__stex_notation_lang_str
2593   \str_clear:N \l__stex_notation_variant_str
2594   \str_clear:N \l__stex_notation_prec_str
2595   \tl_clear:N \l__stex_notation_op_tl
2596
2597   \keys_set:nn { stex / symdef } { #1 }
2598 }
2599
2600 \NewDocumentCommand \symdef { m O{} } {
2601   \__stex_notation_symdef_args:n { #2 }
2602   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2603   \stex_symdecl_do:n { #1 }

```

```

2604 \tl_set:Nn \l_stex_notation_after_do_tl {
2605   \__stex_notation_final:
2606   \stex_smsmode_do:
2607 }
2608 \str_set:Nx \l_stex_get_symbol_uri_str {
2609   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2610 }
2611 \exp_args:Nx \stex_notation_do:nnnnn
2612 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
2613 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2614 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2615 { \l__stex_notation_prec_str}
2616 }
2617 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page [36](#).)

30.3 Variables

```

2618 <@@=stex_variables>
2619
2620 \keys_define:nn { stex / vardef } {
2621   name      .str_set:N = \l__stex_variables_name_str ,
2622   args      .str_set:N = \l__stex_variables_args_str ,
2623   type      .tl_set:N  = \l__stex_variables_type_tl ,
2624   def       .tl_set:N  = \l__stex_variables_def_tl ,
2625   op        .tl_set:N  = \l__stex_variables_op_tl ,
2626   prec      .str_set:N = \l__stex_variables_prec_str ,
2627   assoc     .choices:nn =
2628     {bin,binl,binr,pre,conj,pwconj}
2629     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2630   bind      .choices:nn =
2631     {forall,exists}
2632     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2633 }
2634
2635 \cs_new_protected:Nn \__stex_variables_args:n {
2636   \str_clear:N \l__stex_variables_name_str
2637   \str_clear:N \l__stex_variables_args_str
2638   \str_clear:N \l__stex_variables_prec_str
2639   \str_clear:N \l__stex_variables_assoctype_str
2640   \str_clear:N \l__stex_variables_bind_str
2641   \tl_clear:N \l__stex_variables_type_tl
2642   \tl_clear:N \l__stex_variables_def_tl
2643   \tl_clear:N \l__stex_variables_op_tl
2644
2645   \keys_set:nn { stex / vardef } { #1 }
2646 }
2647
2648 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2649   \__stex_variables_args:n {#2}
2650   \str_if_empty:NT \l__stex_variables_name_str {
2651     \str_set:Nx \l__stex_variables_name_str { #1 }
2652   }

```

```

2653 \prop_clear:N \l_tmpa_prop
2654 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2655
2656 \int_zero:N \l_tmpb_int
2657 \bool_set_true:N \l_tmpa_bool
2658 \str_map_inline:Nn \l__stex_variables_args_str {
2659   \token_case_meaning:NnF ##1 {
2660     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2661     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2662     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2663     {\tl_to_str:n a} {
2664       \bool_set_false:N \l_tmpa_bool
2665       \int_incr:N \l_tmpb_int
2666     }
2667     {\tl_to_str:n B} {
2668       \bool_set_false:N \l_tmpa_bool
2669       \int_incr:N \l_tmpb_int
2670     }
2671   }{
2672     \msg_error:nxxx{stex}{error/wrongargs}{
2673       variable~\l__stex_variables_name_str
2674     }{##1}
2675   }
2676 }
2677 \bool_if:NTF \l_tmpa_bool {
2678   % possibly numeric
2679   \str_if_empty:NTF \l__stex_variables_args_str {
2680     \prop_put:Nnn \l_tmpa_prop { args } {}
2681     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2682   }{
2683     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2684     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2685     \str_clear:N \l_tmpa_str
2686     \int_step_inline:nn \l_tmpa_int {
2687       \str_put_right:Nn \l_tmpa_str i
2688     }
2689     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2690     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2691   }
2692 } {
2693   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2694   \prop_put:Nnx \l_tmpa_prop { arity }
2695   { \str_count:N \l__stex_variables_args_str }
2696 }
2697 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2698 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2699
2700 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2701
2702 \tl_if_empty:NF \l__stex_variables_op_tl {
2703   \cs_set:cpx {
2704     stex_var_op_notation_ \l__stex_variables_name_str _cs
2705   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2706 }

```

```

2707
2708 \tl_set:Nn \l_stex_notation_after_do_tl {
2709   \exp_args:Nne \use:nn {
2710     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2711     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2712   } {{
2713     \exp_after:wN \exp_after:wN \exp_after:wN
2714     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2715     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symb
2716   }}
2717 \stex_if_do_html:T {
2718   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2719     \stex_annotate_invisible:nnn { precedence }
2720     { \l__stex_variables_prec_str }{}
2721     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{\l__stex_variables_type_str}}
2722     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2723     \stex_annotate_invisible:nnn{macroname}{#1}{\l__stex_variables_macroname_str}
2724     \tl_if_empty:NF \l__stex_variables_def_tl {
2725       \stex_annotate_invisible:nnn{definiens}{\l__stex_variables_def_tl}
2726       {\l__stex_variables_def_tl}
2727     }
2728     \str_if_empty:NF \l__stex_variables ASSOCTYPE_str {
2729       \stex_annotate_invisible:nnn{ASSOCTYPE}{\l__stex_variables ASSOCTYPE_str}{}
2730     }
2731     \int_zero:N \l_tmpa_int
2732     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2733     \tl_clear:N \l_tmpa_tl
2734     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2735       \int_incr:N \l_tmpa_int
2736       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2737       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2738       \str_if_eq:VnTF \l_tmpb_str a {
2739         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2740           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2741           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2742         } }
2743       }{
2744         \str_if_eq:VnTF \l_tmpb_str B {
2745           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2746             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2747             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2748           } }
2749         }{
2750           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2751             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2752           } }
2753         }
2754       }
2755     }}
2756     \stex_annotate_invisible:nnn { notationcomp }{{
2757       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2758       $ \exp_args:Nno \use:nn { \use:c {
2759         stex_var_notation_\l__stex_variables_name_str_cs
2760       } } { \l_tmpa_tl } $

```

```

2761     }
2762   }
2763 }
2764 }
2765
2766 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2767 }
2768
2769 \cs_new:Nn \_stex_reset:N {
2770   \tl_if_exist:NTF #1 {
2771     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2772   }{
2773     \let \exp_not:N #1 \exp_not:N \undefined
2774   }
2775 }
2776
2777 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2778   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2779   \exp_args:Nnx \use:nn {
2780     % TODO
2781     \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2782       #2
2783     }
2784   }{
2785     \_stex_reset:N \varnot
2786     \_stex_reset:N \vartype
2787     \_stex_reset:N \vardefi
2788   }
2789 }
2790
2791 \NewDocumentCommand \vardef { s } {
2792   \IfBooleanTF#1 {
2793     \__stex_variables_do_complex:nn
2794   }{
2795     \__stex_variables_do_simple:nnn
2796   }
2797 }
2798
2799 \NewDocumentCommand \svar { 0{} m }{
2800   \tl_if_empty:nTF {#1}{
2801     \str_set:Nn \l_tmpa_str { #2 }
2802   }{
2803     \str_set:Nn \l_tmpa_str { #1 }
2804   }
2805   \_stex_term_omv:nn {
2806     var://\l_tmpa_str
2807   }{ \comp{ #2 } }
2808 }
2809
2810
2811
2812 \keys_define:nn { stex / varseq } {
2813   name      .str_set_x:N = \l__stex_variables_name_str ,
2814   args      .int_set:N   = \l__stex_variables_args_int ,

```

```

2815 type .tl_set:N = \l__stex_variables_type_tl ,
2816 mid .tl_set:N = \l__stex_variables_mid_tl ,
2817 bind .choices:nn =
2818 {forall,exists}
2819 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2820 }
2821
2822 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2823 \str_clear:N \l__stex_variables_name_str
2824 \int_set:Nn \l__stex_variables_args_int 1
2825 \tl_clear:N \l__stex_variables_type_tl
2826 \str_clear:N \l__stex_variables_bind_str
2827
2828 \keys_set:nn { stex / varseq } { #1 }
2829 }
2830
2831 \NewDocumentCommand \varseq {m O{} m m m}{
2832 \__stex_variables_seq_args:n { #2 }
2833 \str_if_empty:NT \l__stex_variables_name_str {
2834 \str_set:Nx \l__stex_variables_name_str { #1 }
2835 }
2836 \prop_clear:N \l_tmpa_prop
2837 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2838
2839 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2840 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2841 \msg_error:nnxx{stex}{error/seqlength}
2842 {\int_use:N \l__stex_variables_args_int}
2843 {\seq_count:N \l_tmpa_seq}
2844 }
2845 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2846 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2847 \msg_error:nnxx{stex}{error/seqlength}
2848 {\int_use:N \l__stex_variables_args_int}
2849 {\seq_count:N \l_tmpb_seq}
2850 }
2851 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2852 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2853
2854 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
2855 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
2856
2857 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2858 \int_step_inline:nn \l__stex_variables_args_int {
2859 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2860 }
2861 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2862 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2863 \tl_if_empty:NF \l__stex_variables_mid_tl {
2864 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2865 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2866 }
2867 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
2868 \int_step_inline:nn \l__stex_variables_args_int {

```

```

2869 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2870 }
2871 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2872 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2873
2874
2875 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2876
2877 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2878
2879 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2880
2881 \int_step_inline:nn \l__stex_variables_args_int {
2882   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2883     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2884   }}
2885 }
2886
2887 \tl_set:Nx \l_tmpa_tl {
2888   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
2889     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2890   }
2891 }
2892
2893 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2894
2895 \exp_args:Nno \use:nn {
2896   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2897   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
2898
2899   \stex_debug:nn{sequences}{New~Sequence:~
2900     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2901     \prop_to_keyval:N \l_tmpa_prop
2902   }
2903
2904   \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2905 }
2906
2907 </package>

```

Chapter 31

STEX -Terms Implementation

```
2908 <*package>
2909
2910 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2911
2912 <@@=stex_terms>
2913
2914 Warnings and error messages
2915 \msg_new:nnn{stex}{error/nonotation}{
2916   Symbol~#1~invoked,~but~has~no~notation~#2!
2917 }
2918 \msg_new:nnn{stex}{error/notationarg}{
2919   Error~in~parsing~notation~#1
2920 }
2921 \msg_new:nnn{stex}{error/noop}{
2922   Symbol~#1~has~no~operator~notation~for~notation~#2
2923 }
2924 \msg_new:nnn{stex}{error/notallowed}{
2925   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2926 }
```

31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2926
2927
2928 \bool_new:N \l_stex_allow_semantic_bool
2929 \bool_set_true:N \l_stex_allow_semantic_bool
2930
2931 \cs_new_protected:Nn \stex_invoke_symbol:n {
2932   \bool_if:NTF \l_stex_allow_semantic_bool {
2933     \str_if_eq:eeF {
2934       \prop_item:cn {
2935         l_stex_symdecl_#1_prop
2936       }{ deprecate }
2937     }
2938   }
```



```

2937   }{}{
2938     \msg_warning:nxxx{stex}{warning/deprecated}{
2939       Symbol~#1
2940     }{
2941       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2942     }
2943   }
2944   \if_mode_math:
2945     \exp_after:wN \__stex_terms_invoke_math:n
2946   \else:
2947     \exp_after:wN \__stex_terms_invoke_text:n
2948   \fi: { #1 }
2949 }{
2950   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2951 }
2952 }
2953
2954 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2955   \peek_charcode_remove:NTF ! {
2956     \__stex_terms_invoke_op_custom:nn {#1}
2957   }{
2958     \__stex_terms_invoke_custom:nn {#1}
2959   }
2960 }
2961
2962 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2963   \peek_charcode_remove:NTF ! {
2964     % operator
2965     \peek_charcode_remove:NTF * {
2966       % custom op
2967       \__stex_terms_invoke_op_custom:nn {#1}
2968     }{
2969       % op notation
2970       \peek_charcode:NTF [ {
2971         \__stex_terms_invoke_op_notation:nw {#1}
2972       }{
2973         \__stex_terms_invoke_op_notation:nw {#1}[]
2974       }
2975     }
2976   }{
2977     \peek_charcode_remove:NTF * {
2978       \__stex_terms_invoke_custom:nn {#1}
2979       % custom
2980     }{
2981       % normal
2982       \peek_charcode:NTF [ {
2983         \__stex_terms_invoke_notation:nw {#1}
2984       }{
2985         \__stex_terms_invoke_notation:nw {#1}[]
2986       }
2987     }
2988   }
2989 }
2990

```

```

2991
2992 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2993   \exp_args:Nnx \use:nn {
2994     \def\comp{\_comp}
2995     \str_set:Nn \l_stex_current_symbol_str { #1 }
2996     \bool_set_false:N \l_stex_allow_semantic_bool
2997     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2998       \comp{ #2 }
2999     }
3000   }{
3001     \stex_reset:N \comp
3002     \stex_reset:N \l_stex_current_symbol_str
3003     \bool_set_true:N \l_stex_allow_semantic_bool
3004   }
3005 }
3006
3007 \keys_define:nn { stex / terms } {
3008   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3009   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3010   unknown .code:n = \str_set:Nx
3011     \l_stex_notation_variant_str \l_keys_key_str
3012 }
3013
3014 \cs_new_protected:Nn \__stex_terms_args:n {
3015   \str_clear:N \l_stex_notation_lang_str
3016   \str_clear:N \l_stex_notation_variant_str
3017
3018   \keys_set:nn { stex / terms } { #1 }
3019 }
3020
3021 \cs_new_protected:Nn \stex_find_notation:nn {
3022   \__stex_terms_args:n { #2 }
3023   \seq_if_empty:cTF {
3024     l_stex_symdecl_ #1 _notations
3025   } {
3026     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3027   } {
3028     \bool_lazy_all:nTF {
3029       {\str_if_empty_p:N \l_stex_notation_variant_str}
3030       {\str_if_empty_p:N \l_stex_notation_lang_str}
3031     }{
3032       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3033     }{
3034       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3035         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3036       }{
3037         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3038       }{
3039         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3040           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3041         }
3042       }
3043     }
3044   }

```

```

3045 }
3046
3047 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3048   \exp_args:Nnx \use:nn {
3049     \def\comp{\_comp}
3050     \str_set:Nn \l_stex_current_symbol_str { #1 }
3051     \stex_find_notation:nn { #1 }{ #2 }
3052     \bool_set_false:N \l_stex_allow_semantic_bool
3053     \cs_if_exist:cTF {
3054       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3055     }{
3056       \_stex_term_oms:nnn {
3057         #1 \c_hash_str \l_stex_notation_variant_str
3058       }{ #1 }{
3059         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3060       }
3061     }{
3062       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3063         \cs_if_exist:cTF {
3064           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3065         }{
3066           \tl_set:Nx \stex_symbol_after_invokation_tl {
3067             \_stex_reset:N \comp
3068             \_stex_reset:N \stex_symbol_after_invokation_tl
3069             \_stex_reset:N \l_stex_current_symbol_str
3070             \bool_set_true:N \l_stex_allow_semantic_bool
3071           }
3072           \def\comp{\_comp}
3073           \str_set:Nn \l_stex_current_symbol_str { #1 }
3074           \bool_set_false:N \l_stex_allow_semantic_bool
3075           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3076         }{
3077           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3078             ~\l_stex_notation_variant_str
3079           }
3080         }
3081       }{
3082         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3083       }
3084     }
3085   }{
3086     \_stex_reset:N \comp
3087     \_stex_reset:N \l_stex_current_symbol_str
3088     \bool_set_true:N \l_stex_allow_semantic_bool
3089   }
3090 }
3091
3092 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3093   \stex_find_notation:nn { #1 }{ #2 }
3094   \cs_if_exist:cTF {
3095     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3096   }{
3097     \tl_set:Nx \stex_symbol_after_invokation_tl {
3098       \_stex_reset:N \comp

```

```

3099     \stex_reset:N \stex_symbol_after_invokation_tl
3100     \stex_reset:N \l_stex_current_symbol_str
3101     \bool_set_true:N \l_stex_allow_semantic_bool
3102   }
3103   \def\comp{\_comp}
3104   \str_set:Nn \l_stex_current_symbol_str { #1 }
3105   \bool_set_false:N \l_stex_allow_semantic_bool
3106   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3107 }{
3108   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3109     ~\l_stex_notation_variant_str
3110   }
3111 }
3112 }
3113
3114 \prop_new:N \l__stex_terms_custom_args_prop
3115
3116 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3117   \exp_args:Nnx \use:nn {
3118     \bool_set_false:N \l_stex_allow_semantic_bool
3119     \def\comp{\_comp}
3120     \str_set:Nn \l_stex_current_symbol_str { #1 }
3121     \prop_clear:N \l__stex_terms_custom_args_prop
3122     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3123     \prop_get:cnN {
3124       l_stex_symdecl_#1 _prop
3125     }{ args } \l_tmpa_str
3126     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3127     \tl_set:Nn \arg { \__stex_terms_arg: }
3128     \str_if_empty:NTF \l_tmpa_str {
3129       \stex_term_oms:nnn {#1}{#1}{#2}
3130     }{
3131       \str_if_in:NnTF \l_tmpa_str b {
3132         \stex_term_ombind:nnn {#1}{#1}{#2}
3133       }{
3134         \str_if_in:NnTF \l_tmpa_str B {
3135           \stex_term_ombind:nnn {#1}{#1}{#2}
3136         }{
3137           \stex_term_oma:nnn {#1}{#1}{#2}
3138         }
3139       }
3140     }
3141     % TODO check that all arguments exist
3142   }{
3143     \stex_reset:N \l_stex_current_symbol_str
3144     \stex_reset:N \arg
3145     \stex_reset:N \comp
3146     \stex_reset:N \l__stex_terms_custom_args_prop
3147     \bool_set_true:N \l_stex_allow_semantic_bool
3148   }
3149 }
3150
3151 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3152   \tl_if_empty:nTF {#2}{

```

```

3153 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3154 \bool_set_true:N \l_tmpa_bool
3155 \bool_do_while:Nn \l_tmpa_bool {
3156   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3157   \int_incr:N \l_tmpa_int
3158 }{
3159   \bool_set_false:N \l_tmpa_bool
3160 }
3161 }
3162 }{
3163   \int_set:Nn \l_tmpa_int { #2 }
3164   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3165     % TODO throw error
3166   }
3167 }
3168 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3169 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3170   % TODO throw error
3171 }
3172 \bool_set_true:N \l_stex_allow_semantic_bool
3173 \IfBooleanTF#1{
3174   \stex_annotate_invisible:n {
3175     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3176   }
3177 }{
3178   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3179 }
3180 \bool_set_false:N \l_stex_allow_semantic_bool
3181 }
3182
3183
3184 \cs_new_protected:Nn \_stex_term_arg:nn {
3185   \bool_set_true:N \l_stex_allow_semantic_bool
3186   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3187   \bool_set_false:N \l_stex_allow_semantic_bool
3188 }
3189
3190 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3191   \exp_args:Nnx \use:nn
3192   { \int_set:Nn \l__stex_terms_downprec { #2 }
3193     \_stex_term_arg:nn { #1 }{ #3 }
3194   }
3195   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3196 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3197 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3198   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3199   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3200   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3201     \expandafter\if\expandafter\relax\noexpand#3
3202     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```

```

3203 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3204 }{
3205 \__stex_terms_math_assoc_arg_simple:n{#3}
3206 }
3207 }
3208
3209 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3210 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3211 \str_if_empty:NTF \l_tmpa_str {
3212 \exp_args:Nx \cs_if_eq:NNTF {
3213 \tl_head:N #1
3214 } \stex_invoke_sequence:n {
3215 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3216 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3217 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3218 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3219 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3220 \exp_not:n{\exp_args:Nnx \use:nn} {
3221 \exp_not:n {
3222 \def\comp{\_varcomp}
3223 \str_set:Nn \l_stex_current_symbol_str
3224 } {varseq://\l_tmpa_str}
3225 \exp_not:n{ ##1 }
3226 }{
3227 \exp_not:n {
3228 \_stex_reset:N \comp
3229 \_stex_reset:N \l_stex_current_symbol_str
3230 }
3231 }
3232 }}}
3233 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3234 \seq_reverse:N \l_tmpa_seq
3235 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3236 \seq_map_inline:Nn \l_tmpa_seq {
3237 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3238 \exp_args:Nno
3239 \l_tmpa_cs { ##1 } \l_tmpa_tl
3240 }
3241 }
3242 \tl_set:Nx \l_tmpa_tl {
3243 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3244 \exp_args:No \exp_not:n \l_tmpa_tl
3245 }
3246 }
3247 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3248 }{
3249 \__stex_terms_math_assoc_arg_simple:n { #1 }
3250 }
3251 } {
3252 \__stex_terms_math_assoc_arg_simple:n { #1 }
3253 }
3254
3255 }
3256

```

```

3257 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3258   \clist_set:Nn \l_tmpa_clist{ #1 }
3259   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3260     \tl_set:Nn \l_tmpa_tl { #1 }
3261   }{
3262     \clist_reverse:N \l_tmpa_clist
3263     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3264
3265     \clist_map_inline:Nn \l_tmpa_clist {
3266       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3267         \exp_args:Nno
3268         \l_tmpa_cs { ##1 } \l_tmpa_tl
3269       }
3270     }
3271   }
3272   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3273 }

```

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

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3274 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3275 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3276 \int_new:N \l__stex_terms_downprec
3277 \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 38.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3278 \tl_set:Nn \l__stex_terms_left_bracket_str (
3279 \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

```

3280 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3281   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3282     \bool_set_false:N \l__stex_terms_brackets_done_bool
3283     #2
3284   } {
3285     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3286       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3287         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3288         \dobrackets { #2 }
3289       }
3290     }{ #2 }
3291   }
3292 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

3293 \bool_new:N \l__stex_terms_brackets_done_bool
3294 %\RequirePackage{scalerel}
3295 \cs_new_protected:Npn \dobrackets #1 {
3296   %\ThisStyle{\if D\m@switch
3297   %   \exp_args:Nnx \use:nn
3298   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3299   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3300   % \else
3301   \exp_args:Nnx \use:nn
3302   {
3303     \bool_set_true:N \l__stex_terms_brackets_done_bool
3304     \int_set:Nn \l__stex_terms_downprec \infpref
3305     \l__stex_terms_left_bracket_str
3306     #1
3307   }
3308   {
3309     \bool_set_false:N \l__stex_terms_brackets_done_bool
3310     \l__stex_terms_right_bracket_str
3311     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3312   }
3313   %\fi}
3314 }

```

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

`\withbrackets`

```

3315 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3316   \exp_args:Nnx \use:nn
3317   {
3318     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3319     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3320     #3
3321   }
3322   {
3323     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3324     {\l__stex_terms_left_bracket_str}
3325     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3326     {\l__stex_terms_right_bracket_str}
3327   }
3328 }

```

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

`\STEXinvisible`

```

3329 \cs_new_protected:Npn \STEXinvisible #1 {
3330   \stex_annotate_invisible:n { #1 }
3331 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
3332 \cs_new_protected:Nn \_stex_term_oms:nnn {
3333   \stex_annotate:nnn{ OMID }{ #2 }{
3334     \stex_highlight_term:nn { #1 } { #3 }
3335   }
3336 }
3337
3338 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3339   \__stex_terms_maybe_brackets:nn { #3 }{
3340     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3341   }
3342 }
```

(End definition for _stex_term_math_oms:nnnn. This function is documented on page 37.)

`_stex_term_math_omv:nn`

```
3343 \cs_new_protected:Nn \_stex_term_omv:nn {
3344   \stex_annotate:nnn{ OMV }{ #1 }{
3345     \stex_highlight_term:nn { #1 } { #2 }
3346   }
3347 }
```

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

`_stex_term_math_oma:nnnn`

```
3348 \cs_new_protected:Nn \_stex_term_oma:nnn {
3349   \stex_annotate:nnn{ OMA }{ #2 }{
3350     \stex_highlight_term:nn { #1 } { #3 }
3351   }
3352 }
3353
3354 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3355   \__stex_terms_maybe_brackets:nn { #3 }{
3356     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3357   }
3358 }
```

(End definition for _stex_term_math_oma:nnnn. This function is documented on page 37.)

`_stex_term_math_omb:nnnn`

```
3359 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3360   \stex_annotate:nnn{ OMBIND }{ #2 }{
3361     \stex_highlight_term:nn { #1 } { #3 }
3362   }
3363 }
3364
3365 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3366   \__stex_terms_maybe_brackets:nn { #3 }{
3367     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3368   }
3369 }
```

(End definition for _stex_term_math_omb:nnnn. This function is documented on page 37.)

```

\symref
\symname
3370 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3371
3372 \keys_define:nn { stex / symname } {
3373   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3374   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3375   root     .tl_set_x:N = \l__stex_terms_root_tl
3376 }
3377
3378 \cs_new_protected:Nn \stex_symname_args:n {
3379   \tl_clear:N \l__stex_terms_post_tl
3380   \tl_clear:N \l__stex_terms_pre_tl
3381   \tl_clear:N \l__stex_terms_root_str
3382   \keys_set:nn { stex / symname } { #1 }
3383 }
3384
3385 \NewDocumentCommand \symref { m m }{
3386   \let\compemph_uri_prev:\compemph@uri
3387   \let\compemph@uri\symrefemph@uri
3388   \STEXsymbol{#1}!\{ #2 }
3389   \let\compemph@uri\compemph_uri_prev:
3390 }
3391
3392 \NewDocumentCommand \synonym { 0{} m m }{
3393   \stex_symname_args:n { #1 }
3394   \let\compemph_uri_prev:\compemph@uri
3395   \let\compemph@uri\symrefemph@uri
3396   % TODO
3397   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3398   \let\compemph@uri\compemph_uri_prev:
3399 }
3400
3401 \NewDocumentCommand \symname { 0{} m }{
3402   \stex_symname_args:n { #1 }
3403   \stex_get_symbol:n { #2 }
3404   \str_set:Nx \l_tmpa_str {
3405     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3406   }
3407   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3408
3409   \let\compemph_uri_prev:\compemph@uri
3410   \let\compemph@uri\symrefemph@uri
3411   \exp_args:NNx \use:nn
3412   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3413     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3414   } }
3415   \let\compemph@uri\compemph_uri_prev:
3416 }
3417
3418 \NewDocumentCommand \Symname { 0{} m }{
3419   \stex_symname_args:n { #1 }
3420   \stex_get_symbol:n { #2 }
3421   \str_set:Nx \l_tmpa_str {
3422     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3423 }
3424 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3425 \let\compemph_uri_prev:\compemph@uri
3426 \let\compemph@uri\symrefemph@uri
3427 \exp_args:NNx \use:nn
3428 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3429   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3430   \l__stex_terms_post_tl
3431 } }
3432 \let\compemph@uri\compemph_uri_prev:
3433 }

```

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

31.3 Notation Components

```

3434 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3435 \cs_new_protected:Nn \stex_highlight_term:nn {
3436   #2
3437 }
3438
3439 \cs_new_protected:Nn \stex_unhighlight_term:n {
3440   % \latexml_if:TF {
3441   %   #1
3442   % } {
3443   %   \rustex_if:TF {
3444   %     #1
3445   %   } {
3446   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3447   %   }
3448   % }
3449 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3450 \cs_new_protected:Npn \_comp #1 {
3451   \str_if_empty:NF \l_stex_current_symbol_str {
3452     \rustex_if:TF {
3453       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3454     }{
3455       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3456     }
3457   }
3458 }
3459
3460 \cs_new_protected:Npn \_varcomp #1 {
3461   \str_if_empty:NF \l_stex_current_symbol_str {
3462     \rustex_if:TF {
3463       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3464     }{
3465       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3466     }
3467   }
3468 }
3469
3470 \def\comp{\_comp}
3471
3472 \cs_new_protected:Npn \compemph@uri #1 #2 {
3473   \compemph{ #1 }
3474 }
3475
3476
3477 \cs_new_protected:Npn \compemph #1 {
3478   #1
3479 }
3480
3481 \cs_new_protected:Npn \defemph@uri #1 #2 {
3482   \defemph{#1}
3483 }
3484
3485 \cs_new_protected:Npn \defemph #1 {
3486   \textbf{#1}
3487 }
3488
3489 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3490   \symrefemph{#1}
3491 }
3492
3493 \cs_new_protected:Npn \symrefemph #1 {
3494   \textbf{#1}
3495 }
3496
3497 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3498   \varempemph{#1}
3499 }
3500
3501 \cs_new_protected:Npn \varempemph #1 {
3502   #1
3503 }

```

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

\ellipses

```

3504 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3505 \bool_new:N \l_stex_inarray_bool
\parrayline 3506 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3507 \NewDocumentCommand \parray { m m } {
\parraycell 3508   \begin{group}
3509   \bool_set_true:N \l_stex_inarray_bool
3510   \begin{array}{#1}
3511     #2
3512   \end{array}

```

```

3513 \endgroup
3514 }
3515
3516 \NewDocumentCommand \prmatrix { m } {
3517   \begingroup
3518   \bool_set_true:N \l_stex_inarray_bool
3519   \begin{matrix}
3520     #1
3521   \end{matrix}
3522   \endgroup
3523 }
3524
3525 \def \maybepline {
3526   \bool_if:NT \l_stex_inarray_bool {\hline}
3527 }
3528
3529 \def \parrayline #1 #2 {
3530   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3531 }
3532
3533 \def \pmrow #1 { \parrayline{}{ #1 } }
3534
3535 \def \parraylineh #1 #2 {
3536   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3537 }
3538
3539 \def \parraycell #1 {
3540   #1 \bool_if:NT \l_stex_inarray_bool {&}
3541 }

```

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

31.4 Variables

```

3542 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3543 \cs_new_protected:Nn \stex_invoke_variable:n {
3544   \if_mode_math:
3545     \exp_after:wN \__stex_variables_invoke_math:n
3546   \else:
3547     \exp_after:wN \__stex_variables_invoke_text:n
3548   \fi: {#1}
3549 }
3550
3551 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3552   %TODO
3553 }
3554
3555
3556 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3557   \peek_charcode_remove:NTF ! {
3558     \peek_charcode_remove:NTF ! {
3559       \peek_charcode:NTF [ {

```

```

3560     \__stex_variables_invoke_op_custom:nw
3561   }{
3562     % TODO throw error
3563   }
3564 }{
3565   \__stex_variables_invoke_op:n { #1 }
3566 }
3567 }{
3568   \peek_charcode_remove:NTF * {
3569     \__stex_variables_invoke_text:n { #1 }
3570   }{
3571     \__stex_variables_invoke_math_ii:n { #1 }
3572   }
3573 }
3574 }
3575
3576 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3577   \cs_if_exist:cTF {
3578     stex_var_op_notation_ #1 _cs
3579   }{
3580     \exp_args:Nnx \use:nn {
3581       \def\comp{\_varcomp}
3582       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3583       \_stex_term_omv:nn { var://#1 }{
3584         \use:c{stex_var_op_notation_ #1 _cs }
3585       }
3586     }{
3587       \_stex_reset:N \comp
3588       \_stex_reset:N \l_stex_current_symbol_str
3589     }
3590   }{
3591     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3592       \__stex_variables_invoke_math_ii:n {#1}
3593     }{
3594       \msg_error:nxxx{stex}{error/noop}{variable~#1}{-}
3595     }
3596   }
3597 }
3598
3599 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3600   \cs_if_exist:cTF {
3601     stex_var_notation_#1_cs
3602   }{
3603     \tl_set:Nx \stex_symbol_after_invokation_tl {
3604       \_stex_reset:N \comp
3605       \_stex_reset:N \stex_symbol_after_invokation_tl
3606       \_stex_reset:N \l_stex_current_symbol_str
3607       \bool_set_true:N \l_stex_allow_semantic_bool
3608     }
3609     \def\comp{\_varcomp}
3610     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3611     \bool_set_false:N \l_stex_allow_semantic_bool
3612     \use:c{stex_var_notation_#1_cs}
3613   }{

```

```

3614 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3615 }
3616 }

```

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

31.5 Sequences

```

3617 <@@=stex_sequences>
3618
3619 \cs_new_protected:Nn \stex_invoke_sequence:n {
3620   \peek_charcode_remove:NTF ! {
3621     \stex_term_omv:nn {varseq://#1}{
3622       \exp_args:Nnx \use:nn {
3623         \def\comp{\_varcomp}
3624         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3625         \prop_item:cn{stex_varseq_#1_prop}{notation}
3626       }{
3627         \stex_reset:N \comp
3628         \stex_reset:N \l_stex_current_symbol_str
3629       }
3630     }
3631   }{
3632     \bool_set_false:N \l_stex_allow_semantic_bool
3633     \def\comp{\_varcomp}
3634     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3635     \tl_set:Nx \stex_symbol_after_invokation_tl {
3636       \stex_reset:N \comp
3637       \stex_reset:N \stex_symbol_after_invokation_tl
3638       \stex_reset:N \l_stex_current_symbol_str
3639       \bool_set_true:N \l_stex_allow_semantic_bool
3640     }
3641     \use:c { stex_varseq_#1_cs }
3642   }
3643 }
3644 </package>

```

Chapter 32

STEX -Structural Features Implementation

```
3645 <*package>
3646
3647 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3648
3649
3650 Warnings and error messages
3651 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3652   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3653 }
3654 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3655   Symbol~#1~not~assigned~in~interpretmodule~#2
3656 }
3657 \msg_new:nnn{stex}{error/unknownstructure}{
3658   No~structure~#1~found!
3659 }
3660 \msg_new:nnn{stex}{error/unknownfield}{
3661   No~field~#1~in~instance~#2~found!
3662 }
3663 \msg_new:nnn{stex}{error/keyval}{
3664   Invalid~key=value~pair~#1
3665 }
3666 \msg_new:nnn{stex}{error/instantiate/missing}{
3667   Assignments~missing~in~instantiate:~#1
3668 }
3669 \msg_new:nnn{stex}{error/incompatible}{
3670   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3671 }
3672
3673
```


32.1 Imports with modification

```

3674 <@@=stex_copymodule>
3675 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3676   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3677     \tl_set:Nn \l_tmpa_tl { #1 }
3678     \__stex_copymodule_get_symbol_from_cs:
3679   }{
3680     % argument is a string
3681     % is it a command name?
3682     \cs_if_exist:cTF { #1 }{
3683       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3684       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3685       \str_if_empty:NNTF \l_tmpa_str {
3686         \exp_args:Nx \cs_if_eq:NNTF {
3687           \tl_head:N \l_tmpa_tl
3688         } \stex_invoke_symbol:n {
3689           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3690         }{
3691           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3692         }
3693       } {
3694         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3695       }
3696     }{
3697       % argument is not a command name
3698       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3699       % \l_stex_all_symbols_seq
3700     }
3701   }
3702 }
3703
3704 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3705   \str_set:Nn \l_tmpa_str { #1 }
3706   \bool_set_false:N \l_tmpa_bool
3707   \bool_if:NF \l_tmpa_bool {
3708     \tl_set:Nn \l_tmpa_tl {
3709       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3710     }
3711     \str_set:Nn \l_tmpa_str { #1 }
3712     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3713     \seq_map_inline:Nn #2 {
3714       \str_set:Nn \l_tmpb_str { ##1 }
3715       \str_if_eq:eeT { \l_tmpa_str } {
3716         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3717       } {
3718         \seq_map_break:n {
3719           \tl_set:Nn \l_tmpa_tl {
3720             \str_set:Nn \l_stex_get_symbol_uri_str {
3721               ##1
3722             }
3723           }
3724         }
3725       }

```

```

3726     }
3727     \l_tmpa_tl
3728   }
3729 }
3730
3731 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3732   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3733     { \tl_tail:N \l_tmpa_tl }
3734   \tl_if_single:NTF \l_tmpa_tl {
3735     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3736       \exp_after:wN \str_set:Nn \exp_after:wN
3737         \l_stex_get_symbol_uri_str \l_tmpa_tl
3738       \__stex_copymodule_get_symbol_check:n { #1 }
3739     }{
3740       % TODO
3741       % tail is not a single group
3742     }
3743   }{
3744     % TODO
3745     % tail is not a single group
3746   }
3747 }
3748
3749 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3750   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3751     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3752       :~\seq_use:Nn #1 {,~}
3753     }
3754   }
3755 }
3756
3757 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3758   \stex_import_module_uri:nn { #1 } { #2 }
3759   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3760   \stex_import_require_module:nnnn
3761     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3762     { \l_stex_import_path_str } { \l_stex_import_name_str }
3763   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3764   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3765   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3766   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3767     \seq_map_inline:cn {c_stex_module_###_constants}{
3768       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3769         ##1 ? ####1
3770       }
3771     }
3772   }
3773   \seq_clear:N \l_tmpa_seq
3774   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3775     name      = \l_stex_current_copymodule_name_str ,
3776     module    = \l_stex_current_module_str ,
3777     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3778     includes  = \l_tmpa_seq ,
3779     fields    = \l_tmpa_seq

```

```

3780 }
3781 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3782   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3783   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3784 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3785 \stex_if_smsmode:F {
3786   \begin{stex_annotate_env} {#4} {
3787     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3788   }
3789   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3790 }
3791 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3792 \bool_set_false:N \_stex_html_do_output_bool
3793 }
3794 \cs_new_protected:Nn \stex_copymodule_end:n {
3795   \def \l_tmpa_cs ##1 ##2 {#1}
3796   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3797   \tl_clear:N \l_tmpa_tl
3798   \tl_clear:N \l_tmpb_tl
3799   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3800   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3801     \seq_map_inline:cn {c_stex_module_##1_constants}{
3802       \tl_clear:N \l_tmpc_tl
3803       \l_tmpa_cs{##1}{####1}
3804       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3805         \tl_put_right:Nx \l_tmpa_tl {
3806           \prop_set_from_keyval:cn {
3807             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3808           }{
3809             \exp_after:wN \prop_to_keyval:N \csname
3810               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3811             \endcsname
3812           }
3813           \seq_clear:c {
3814             l_stex_symdecl_
3815             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
3816             _notations
3817           }
3818         }
3819         \tl_put_right:Nx \l_tmpc_tl {
3820           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
3821           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1
3822         }
3823         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
3824         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3825           \tl_put_right:Nx \l_tmpc_tl {
3826             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3827           }
3828           \tl_put_right:Nx \l_tmpa_tl {
3829             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3830               \stex_invoke_symbol:n {
3831                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1
3832             }
3833           }

```

```

3834     }
3835   }
3836 }{
3837   \tl_put_right:Nx \l_tmpc_tl {
3838     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3839   }
3840   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3841   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3842   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3843   \tl_put_right:Nx \l_tmpa_tl {
3844     \prop_set_from_keyval:cn {
3845       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3846     }{
3847       \prop_to_keyval:N \l_tmpa_prop
3848     }
3849     \seq_clear:c {
3850       l_stex_symdecl_
3851       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3852       _notations
3853     }
3854   }
3855   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3856   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3857     \tl_put_right:Nx \l_tmpc_tl {
3858       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3859     }
3860     \tl_put_right:Nx \l_tmpa_tl {
3861       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3862         \stex_invoke_symbol:n {
3863           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3864         }
3865       }
3866     }
3867   }
3868 }
3869 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3870   \tl_put_right:Nx \l_tmpc_tl {
3871     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3872   }
3873 }
3874 \tl_put_right:Nx \l_tmpb_tl {
3875   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3876 }
3877 }
3878 }
3879 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3880 \tl_put_left:Nx \l_tmpa_tl {
3881   \prop_set_from_keyval:cn {
3882     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3883   }{
3884     \prop_to_keyval:N \l_stex_current_copymodule_prop
3885   }
3886 }
3887 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3888 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3889 \exp_args:Nx \stex_do_up_to_module:n {
3890   \exp_args:No \exp_not:n \l_tmpa_tl
3891 }
3892 \l_tmpb_tl
3893 \stex_if_smsmode:F {
3894   \end{stex_annotate_env}
3895 }
3896 }
3897
3898 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3899   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3900   \stex_deactivate_macro:Nn \symdecl {module~environments}
3901   \stex_deactivate_macro:Nn \symdef {module~environments}
3902   \stex_deactivate_macro:Nn \notation {module~environments}
3903   \stex_reactivate_macro:N \assign
3904   \stex_reactivate_macro:N \renamedec1
3905   \stex_reactivate_macro:N \donotcopy
3906   \stex_smsmode_do:
3907 }{
3908   \stex_copymodule_end:n {}
3909 }
3910
3911 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3912   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3913   \stex_deactivate_macro:Nn \symdecl {module~environments}
3914   \stex_deactivate_macro:Nn \symdef {module~environments}
3915   \stex_deactivate_macro:Nn \notation {module~environments}
3916   \stex_reactivate_macro:N \assign
3917   \stex_reactivate_macro:N \renamedec1
3918   \stex_reactivate_macro:N \donotcopy
3919   \stex_smsmode_do:
3920 }{
3921   \stex_copymodule_end:n {
3922     \tl_if_exist:cF {
3923       l__stex_copymodule_copymodule_##1?##2_def_tl
3924     }{
3925       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3926         ##1?##2
3927       }{\l_stex_current_copymodule_name_str}
3928     }
3929   }
3930 }
3931
3932 \NewDocumentCommand \donotcopy { 0{} m}{
3933   \stex_import_module_uri:nn { #1 } { #2 }
3934   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3935   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3936     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
3937     \seq_map_inline:cn {c_stex_module_##1_constants}{
3938       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3939       \bool_lazy_any_p:nT {
3940         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3941         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}

```

```

3942         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3943     }{
3944         % TODO throw error
3945     }
3946 }
3947 }
3948
3949 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3950 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3951 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3952 }
3953
3954 \NewDocumentCommand \assign { m m }{
3955     \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3956     \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3957     \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3958 }
3959
3960 \keys_define:nn { stex / renamedec1 } {
3961     name .str_set_x:N = \l_stex_renamedec1_name_str
3962 }
3963 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
3964     \str_clear:N \l_stex_renamedec1_name_str
3965     \keys_set:nn { stex / renamedec1 } { #1 }
3966 }
3967
3968 \NewDocumentCommand \renamedec1 { O{} m m}{
3969     \__stex_copymodule_renamedec1_args:n { #1 }
3970     \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
3971     \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3972     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3973     \str_if_empty:NTF \l_stex_renamedec1_name_str {
3974         \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3975             \l_stex_get_symbol_uri_str
3976         } }
3977     } {
3978         \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
3979         \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3980         \prop_set_eq:cc {l_stex_symdecl_
3981             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3982             _prop
3983         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3984         \seq_set_eq:cc {l_stex_symdecl_
3985             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3986             _notations
3987         }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3988         \prop_put:cnx {l_stex_symdecl_
3989             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3990             _prop
3991         }{ name }{ \l_stex_renamedec1_name_str }
3992         \prop_put:cnx {l_stex_symdecl_
3993             \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3994             _prop
3995         }{ module }{ \l_stex_current_module_str }

```

```

3996 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
3997 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3998 }
3999 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4000 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4001 } }
4002 }
4003 }
4004
4005 \stex_deactivate_macro:Nn \assign {copymodules}
4006 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4007 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4008
4009
4010 \seq_new:N \l_stex_implicit_morphisms_seq
4011 \NewDocumentCommand \implicitmorphism { 0{} m m}{
4012 \stex_import_module_uri:nn { #1 } { #2 }
4013 \stex_debug:nn{implicits}{
4014 Implicit~morphism:~
4015 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4016 }
4017 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4018 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4019 }{
4020 \msg_error:nnn{stex}{error/conflictingmodules}{
4021 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4022 }
4023 }
4024
4025 % TODO
4026
4027
4028
4029 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4030 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4031 }
4032 }
4033

```

32.2 The feature environment

structural@feature

```

4034 <@@=stex_features>
4035
4036 \NewDocumentEnvironment{structural_feature_module}{ m m m m }{
4037 \stex_if_in_module:F {
4038 \msg_set:nnn{stex}{error/nomodule}{
4039 Structural~Feature~has~to~occur~in~a~module:\\
4040 Feature~#2~of~type~#1\\
4041 In~File::~\stex_path_to_string:N \g_stex_currentfile_seq
4042 }
4043 \msg_error:nn{stex}{error/nomodule}
4044 }

```

```

4045
4046 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4047
4048 \stex_if_smsmode:F {
4049   \begin{stex_annotate_env}{ feature:#1 }{}
4050   \stex_annotate_invisible:nnn{header}{}{ #3 }
4051 }
4052 }{
4053 \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4054 \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4055 \stex_debug:nn{features}{
4056   Feature: \l_stex_last_feature_str
4057 }
4058 \stex_if_smsmode:F {
4059   \end{stex_annotate_env}
4060 }
4061 }

```

32.3 Structure

structure

```

4062 <@=stex_structures>
4063 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4064   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4065     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4066   }
4067   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4068     {#1}{#2}
4069 }
4070
4071 \keys_define:nn { stex / features / structure } {
4072   name .str_set_x:N = \l__stex_structures_name_str ,
4073 }
4074
4075 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4076   \str_clear:N \l__stex_structures_name_str
4077   \keys_set:nn { stex / features / structure } { #1 }
4078 }
4079
4080 \NewDocumentEnvironment{mathstructure}{m O{}}{
4081   \__stex_structures_structure_args:n { #2 }
4082   \str_if_empty:NT \l__stex_structures_name_str {
4083     \str_set:Nx \l__stex_structures_name_str { #1 }
4084   }
4085   \exp_args:Nnnx
4086   \begin{structural_feature_module}{ structure }
4087     { \l__stex_structures_name_str }{}
4088   \stex_smsmode_do:
4089 }{
4090   \end{structural_feature_module}
4091   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4092   \seq_clear:N \l_tmpa_seq
4093   \seq_map_inline:Nn \l_stex_collect_imports_seq {

```



```

4094     \seq_map_inline:cn{c_stex_module_##1_constants}{
4095       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4096     }
4097   }
4098   \exp_args:Nnno
4099   \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4100   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4101   \stex_add_structure_to_current_module:nn
4102     \l__stex_structures_name_str
4103     \l_stex_last_feature_str
4104   \exp_args:Nx \stex_symdecl_do:nn {
4105     name = \l__stex_structures_name_str ,
4106     type = \metacollection ,
4107     def = {\STEXsymbol{module-type}{
4108       \_stex_term_math_oms:nnnn { \l_stex_last_feature_str }{}{0}{}}
4109     }}
4110   }{ #1 }
4111   \exp_args:Nx
4112   \stex_add_to_current_module:n {
4113     \tl_set:cn { #1 }{
4114       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4115     }
4116   }
4117   \exp_args:Nx
4118   \stex_do_up_to_module:n {
4119     \tl_set:cn { #1 }{
4120       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4121     }
4122   }
4123 }
4124 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4125
4126 \cs_new:Nn \stex_invoke_structure:nn {
4127   \stex_invoke_symbol:n { #1?#2 }
4128 }
4129
4130 \cs_new_protected:Nn \stex_get_structure:n {
4131   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4132     \tl_set:Nn \l_tmpa_tl { #1 }
4133     \__stex_structures_get_from_cs:
4134   }{
4135     \cs_if_exist:cTF { #1 }{
4136       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4137       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4138       \str_if_empty:NNTF \l_tmpa_str {
4139         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4140           \__stex_structures_get_from_cs:
4141         }{
4142           \__stex_structures_get_from_string:n { #1 }
4143         }
4144       }{
4145         \__stex_structures_get_from_string:n { #1 }
4146       }
4147     }{

```

```

4148     \__stex_structures_get_from_string:n { #1 }
4149   }
4150 }
4151 }
4152
4153 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4154   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4155     { \tl_tail:N \l_tmpa_tl }
4156   \str_set:Nx \l_tmpa_str {
4157     \exp_after:wN \use_i:nn \l_tmpa_tl
4158   }
4159   \str_set:Nx \l_tmpb_str {
4160     \exp_after:wN \use_ii:nn \l_tmpa_tl
4161   }
4162   \str_set:Nx \l_stex_get_structure_str {
4163     \l_tmpa_str ? \l_tmpb_str
4164   }
4165   \str_set:Nx \l_stex_get_structure_module_str {
4166     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4167   }
4168 }
4169
4170 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4171   \tl_set:Nn \l_tmpa_tl {
4172     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4173   }
4174   \str_set:Nn \l_tmpa_str { #1 }
4175   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4176
4177   \seq_map_inline:Nn \l_stex_all_modules_seq {
4178     \prop_if_exist:cT {c_stex_module_##1_structures} {
4179       \prop_map_inline:cn {c_stex_module_##1_structures} {
4180         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4181           \prop_map_break:n{\seq_map_break:n{
4182             \tl_set:Nn \l_tmpa_tl {
4183               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4184               \str_set:Nn \l_stex_get_structure_module_str {####2}
4185             }
4186           }}
4187         }
4188       }
4189     }
4190   }
4191   \l_tmpa_tl
4192 }

```

\instantiate

```

4193
4194 \keys_define:nn { stex / instantiate } {
4195   name .str_set_x:N = \l__stex_structures_name_str
4196 }
4197 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4198   \str_clear:N \l__stex_structures_name_str
4199   \keys_set:nn { stex / instantiate } { #1 }

```

```

4200 }
4201
4202 \NewDocumentCommand \instantiate {m O{} m m}{
4203   \beginingroup
4204     \stex_get_structure:n {#4}
4205     \__stex_structures_instantiate_args:n { #2 }
4206     \str_if_empty:NT \l__stex_structures_name_str {
4207       \str_set:Nn \l__stex_structures_name_str { #1 }
4208     }
4209     \seq_clear:N \l__stex_structures_fields_seq
4210     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4211     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4212       \seq_map_inline:cn {c_stex_module_##1_constants}{
4213         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4214       }
4215     }
4216     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4217     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4218     \prop_clear:N \l_tmpa_prop
4219     \seq_map_inline:Nn \l_tmpa_seq {
4220       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4221       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4222         \msg_error:nnn{stex}{error/keyval}{##1}
4223       }
4224       \exp_args:Nx \stex_get_symbol_in_seq:nn { \seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4225       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4226       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4227       \exp_args:Nx \stex_get_symbol:n { \seq_item:Nn \l_tmpb_seq 2}
4228       \exp_args:Nxx \str_if_eq:nnF
4229         { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4230         { \prop_item:cn{l_stex_symdecl\l__stex_get_symbol_uri_str _prop}{args}}{
4231         \msg_error:nnxxx{stex}{error/incompatible}
4232         { \l__stex_structures_dom_str
4233           { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4234           { \l_stex_get_symbol_uri_str
4235             { \prop_item:cn{l_stex_symdecl\l__stex_get_symbol_uri_str _prop}{args}}
4236           }
4237         \prop_put:Nxx \l_tmpa_prop { \seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4238       }
4239       \seq_if_empty:NF \l__stex_structures_fields_seq {
4240         \msg_error:nnx{stex}{error/instantiate/missing}{ \seq_use:Nn \l__stex_structures_fields_
4241       }
4242       \exp_args:Nx
4243       \stex_add_to_current_module:n {
4244         \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4245         domain = \l_stex_get_structure_module_str ,
4246         \prop_to_keyval:N \l_tmpa_prop
4247       }
4248       \tl_set:cn{ #1 }{ \stex_invoke_instance:nn{ \l_stex_current_module_str?\l__stex_structur
4249     }
4250     \exp_args:Nx
4251     \stex_do_up_to_module:n {
4252       \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4253       domain = \l_stex_get_structure_module_str ,

```

```

4254     \prop_to_keyval:N \l_tmpa_prop
4255   }
4256   \tl_set:cn{ #1 }{\stex_invoke_instance:nn{\l_stex_current_module_str?\l_stex_structur
4257 }
4258 \exp_args:Nxx \stex_symdecl_do:nn {
4259   type={\STEXsymbol{module-type}}{
4260     \stex_term_math_oms:nnnn {
4261       \l_stex_get_structure_module_str
4262     }{}{0}{}
4263   }}
4264   }{\l__stex_structures_name_str}
4265 \endgroup
4266 \stex_smsmode_do:
4267 }
4268 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4269
4270 \cs_new_protected:Nn \stex_symbol_or_var:n {
4271   \cs_if_exist:cTF{#1}{
4272     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4273     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4274     \str_if_empty:NTF \l_tmpa_str {
4275       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4276       \stex_invoke_variable:n {
4277         \bool_set_true:N \l_stex_symbol_or_var_bool
4278         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4279         \str_set:Nx \l_stex_get_symbol_uri_str {
4280           \exp_after:wN \use:n \l_tmpa_tl
4281         }
4282       }{
4283         \bool_set_false:N \l_stex_symbol_or_var_bool
4284         \stex_get_symbol:n{#1}
4285       }
4286     }{
4287       \__stex_structures_symbolorvar_from_string:n{ #1 }
4288     }
4289   }{
4290     \__stex_structures_symbolorvar_from_string:n{ #1 }
4291   }
4292 }
4293
4294 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4295   \prop_if_exist:cTF {\l_stex_variable_#1 _prop}{
4296     \bool_set_true:N \l_stex_symbol_or_var_bool
4297     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4298   }{
4299     \bool_set_false:N \l_stex_symbol_or_var_bool
4300     \stex_get_symbol:n{#1}
4301   }
4302 }
4303
4304
4305 \NewDocumentCommand \varinstantiate {m O{} m m}{
4306   \begingroup
4307     \stex_get_structure:n {#4}

```

```

4308 \__stex_structures_instantiate_args:n { #2 }
4309 \str_if_empty:NT \l__stex_structures_name_str {
4310   \str_set:Nn \l__stex_structures_name_str { #1 }
4311 }
4312 \seq_clear:N \l__stex_structures_fields_seq
4313 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4314 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4315   \seq_map_inline:cn {c_stex_module_##1_constants}{
4316     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4317   }
4318 }
4319 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4320 \prop_clear:N \l_tmpa_prop
4321 \tl_if_empty:nF {#3} {
4322   \seq_set_split:Nnn \l_tmpa_seq , {#3}
4323   \seq_map_inline:Nn \l_tmpa_seq {
4324     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4325     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4326       \msg_error:nnn{stex}{error/keyval}{##1}
4327     }
4328     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4329     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4330     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4331     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4332     \bool_if:NTF \l_stex_symbol_or_var_bool {
4333       \exp_args:Nxx \str_if_eq:nnF
4334         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4335         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4336         \msg_error:nnxxx{stex}{error/incompatible}
4337         {\l__stex_structures_dom_str
4338         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4339         {\l_stex_get_symbol_uri_str
4340         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4341       }
4342       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4343     }{
4344       \exp_args:Nxx \str_if_eq:nnF
4345         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4346         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4347         \msg_error:nnxxx{stex}{error/incompatible}
4348         {\l__stex_structures_dom_str
4349         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4350         {\l_stex_get_symbol_uri_str
4351         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4352       }
4353       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4354     }
4355   }
4356 }
4357 \tl_gclear:N \g__stex_structures_aftergroup_tl
4358 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4359   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4360   \stex_find_notation:nn{##1}{}
4361   \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}

```

```

4362     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4363 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4364     \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}
4365     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4366 }
4367
4368 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4369     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4370         name = \l_tmpa_str ,
4371         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4372         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4373         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4374     }
4375     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4376     {g__stex_structures_tmpa_\l_tmpa_str _cs}
4377     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4378     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4379 }
4380 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invoke
4381 }
4382 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4383     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4384         domain = \l_stex_get_structure_module_str ,
4385         \prop_to_keyval:N \l_tmpa_prop
4386     }
4387     \tl_set:cn { #1 }{\stex_invoke_varinstance:nn {\l__stex_structures_name_str}}
4388 }
4389 \aftergroup\g__stex_structures_aftergroup_tl
4390 \endgroup
4391 \stex_smsmode_do:
4392 }
4393
4394 \cs_new_protected:Nn \stex_invoke_instance:nn {
4395     \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4396         \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4397     }{
4398         \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4399     }
4400 }
4401
4402 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4403     \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4404         \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4405         \l_tmpa_tl
4406     }{
4407         \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4408     }
4409 }

```

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

\stex_invoke_structure:nnn

```

4410 % #1: URI of the instance
4411 % #2: URI of the instantiated module

```

```

4412 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4413   \tl_if_empty:nTF{ #3 }{
4414     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4415       c_stex_feature_ #2 _prop
4416     }
4417     \tl_clear:N \l_tmpa_tl
4418     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4419     \seq_map_inline:Nn \l_tmpa_seq {
4420       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4421       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4422       \cs_if_exist:cT {
4423         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4424       }{
4425         \tl_if_empty:NF \l_tmpa_tl {
4426           \tl_put_right:Nn \l_tmpa_tl {,}
4427         }
4428         \tl_put_right:Nx \l_tmpa_tl {
4429           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4430         }
4431       }
4432     }
4433     \exp_args:No \mathstrut \l_tmpa_tl
4434   }{
4435     \stex_invoke_symbol:n{#1/#3}
4436   }
4437 }

```

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

```

4438 </package>

```

Chapter 33

STEX -Statements Implementation

```
4439 <*package>
4440
4441 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4442
4443 <@@=stex_statements>
    Warnings and error messages
4444
\titleemph
4445 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
4446 \keys_define:nn {stex / definiendum }{
4447   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4448   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4449   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4450   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4451 }
4452 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4453   \str_clear:N \l__stex_statements_definiendum_root_str
4454   \tl_clear:N \l__stex_statements_definiendum_post_tl
4455   \str_clear:N \l__stex_statements_definiendum_gfa_str
4456   \keys_set:nn { stex / definiendum }{ #1 }
4457 }
4458 \NewDocumentCommand \definiendum { O{} m m } {
4459   \__stex_statements_definiendum_args:n { #1 }
4460   \stex_get_symbol:n { #2 }
4461   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4462   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4463     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```



```

4464     \tl_set:Nn \l_tmpa_tl { #3 }
4465   } {
4466     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4467     \tl_set:Nn \l_tmpa_tl {
4468       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4469     }
4470   }
4471 } {
4472   \tl_set:Nn \l_tmpa_tl { #3 }
4473 }
4474
4475 % TODO root
4476 \rustex_if:TF {
4477   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4478 } {
4479   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4480 }
4481 }
4482 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4483
4484 \NewDocumentCommand \definame { 0{ } m } {
4485   \__stex_statements_definiendum_args:n { #1 }
4486   % TODO: root
4487   \stex_get_symbol:n { #2 }
4488   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4489   \str_set:Nx \l_tmpa_str {
4490     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4491   }
4492   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4493   \rustex_if:TF {
4494     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4495       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4496     }
4497   } {
4498     \defemph@uri {
4499       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4500     } { \l_stex_get_symbol_uri_str }
4501   }
4502 }
4503 \stex_deactivate_macro:Nn \definame {definition~environments}
4504
4505 \NewDocumentCommand \Definame { 0{ } m } {
4506   \__stex_statements_definiendum_args:n { #1 }
4507   \stex_get_symbol:n { #2 }
4508   \str_set:Nx \l_tmpa_str {
4509     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4510   }
4511   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4512   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4513   \rustex_if:TF {

```

```

4514 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4515 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4516 }
4517 } {
4518 \defemph@uri {
4519 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4520 } { \l_stex_get_symbol_uri_str }
4521 }
4522 }
4523 \stex_deactivate_macro:Nn \Definame {definition~environments}
4524
4525 \NewDocumentCommand \premise { m }{
4526 \stex_annotate:nnn{ premise }{}{ #1 }
4527 }
4528 \NewDocumentCommand \conclusion { m }{
4529 \stex_annotate:nnn{ conclusion }{}{ #1 }
4530 }
4531 \NewDocumentCommand \definiens { m }{
4532 \stex_annotate:nnn{ definiens }{}{ #1 }
4533 }
4534
4535 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4536 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4537 \stex_deactivate_macro:Nn \definiens {definition~environments}
4538

```

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

sdefinition

```

4539
4540 \keys_define:nn {stex / sdefinition }{
4541 type .str_set_x:N = \sdefinitiontype,
4542 id .str_set_x:N = \sdefinitionid,
4543 name .str_set_x:N = \sdefinitionname,
4544 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4545 title .tl_set:N = \sdefinitiontitle
4546 }
4547 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4548 \str_clear:N \sdefinitiontype
4549 \str_clear:N \sdefinitionid
4550 \str_clear:N \sdefinitionname
4551 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4552 \tl_clear:N \sdefinitiontitle
4553 \keys_set:nn { stex / sdefinition }{ #1 }
4554 }
4555
4556 \NewDocumentEnvironment{sdefinition}{0{}}{
4557 \__stex_statements_sdefinition_args:n{ #1 }
4558 \stex_reactivate_macro:N \definiendum
4559 \stex_reactivate_macro:N \definame
4560 \stex_reactivate_macro:N \Definame
4561 \stex_reactivate_macro:N \premise
4562 \stex_reactivate_macro:N \definiens
4563 \stex_if_smsmode:F{

```

```

4564 \seq_clear:N \l_tmpa_seq
4565 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4566   \tl_if_empty:nF{ ##1 }{
4567     \stex_get_symbol:n { ##1 }
4568     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4569       \l_stex_get_symbol_uri_str
4570     }
4571   }
4572 }
4573 \exp_args:Nnnx
4574 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4575 \str_if_empty:NF \sdefinitiontype {
4576   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4577 }
4578 \clist_set:No \l_tmpa_clist \sdefinitiontype
4579 \tl_clear:N \l_tmpa_tl
4580 \clist_map_inline:Nn \l_tmpa_clist {
4581   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4582     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4583   }
4584 }
4585 \tl_if_empty:NTF \l_tmpa_tl {
4586   \__stex_statements_sdefinition_start:
4587 }{
4588   \l_tmpa_tl
4589 }
4590 }
4591 \stex_ref_new_doc_target:n \sdefinitionid
4592 \stex_smsmode_do:
4593 }{
4594   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4595 \stex_if_smsmode:F {
4596   \clist_set:No \l_tmpa_clist \sdefinitiontype
4597   \tl_clear:N \l_tmpa_tl
4598   \clist_map_inline:Nn \l_tmpa_clist {
4599     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4600       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4601     }
4602   }
4603   \tl_if_empty:NTF \l_tmpa_tl {
4604     \__stex_statements_sdefinition_end:
4605   }{
4606     \l_tmpa_tl
4607   }
4608   \end{stex_annotate_env}
4609 }
4610 }

```

\stexpatchdefinition

```

4611 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4612   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4613     ~(\sdefinitiontitle)
4614   }~}
4615 }

```

```

4616 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4617
4618 \newcommand\stexpatchdefinition[3] [] {
4619   \str_set:Nx \l_tmpa_str{ #1 }
4620   \str_if_empty:NTF \l_tmpa_str {
4621     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4622     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4623   }{
4624     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4625     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4626   }
4627 }

```

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

\inlinedef inline:

```

4628 \keys_define:nn {stex / inlinedef }{
4629   type      .str_set_x:N = \sdefinitiontype,
4630   id        .str_set_x:N = \sdefinitionid,
4631   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4632   name      .str_set_x:N = \sdefinitionname
4633 }
4634 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4635   \str_clear:N \sdefinitiontype
4636   \str_clear:N \sdefinitionid
4637   \str_clear:N \sdefinitionname
4638   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4639   \keys_set:nn { stex / inlinedef }{ #1 }
4640 }
4641 \NewDocumentCommand \inlinedef { 0{} m } {
4642   \beginngroup
4643   \__stex_statements_inlinedef_args:n{ #1 }
4644   \stex_reactivate_macro:N \definiendum
4645   \stex_reactivate_macro:N \definame
4646   \stex_reactivate_macro:N \Definame
4647   \stex_reactivate_macro:N \premise
4648   \stex_reactivate_macro:N \definiens
4649   \stex_ref_new_doc_target:n \sdefinitionid
4650   \stex_if_smsmode:TF{
4651     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4652   }{
4653     \seq_clear:N \l_tmpa_seq
4654     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4655       \tl_if_empty:nF{ ##1 }{
4656         \stex_get_symbol:n { ##1 }
4657         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4658           \l_stex_get_symbol_uri_str
4659         }
4660       }
4661     }
4662     \exp_args:Nnx
4663     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4664       \str_if_empty:NF \sdefinitiontype {
4665         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```

```

4666     }
4667     #2
4668     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4669   }
4670 }
4671 \endgroup
4672 \stex_smsmode_do:
4673 }

```

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

33.2 Assertions

`sassertion`

```

4674
4675 \keys_define:nn {stex / sassertion }{
4676   type      .str_set_x:N = \sassertiontype,
4677   id        .str_set_x:N = \sassertionid,
4678   title     .tl_set:N    = \sassertiontitle ,
4679   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4680   name      .str_set_x:N = \sassertionname
4681 }
4682 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4683   \str_clear:N \sassertiontype
4684   \str_clear:N \sassertionid
4685   \str_clear:N \sassertionname
4686   \clist_clear:N \l__stex_statements_sassertion_for_clist
4687   \tl_clear:N \sassertiontitle
4688   \keys_set:nn { stex / sassertion }{ #1 }
4689 }
4690
4691 %\tl_new:N \g__stex_statements_aftergroup_tl
4692
4693 \NewDocumentEnvironment{sassertion}{0{}}{
4694   \__stex_statements_sassertion_args:n{ #1 }
4695   \stex_reactivate_macro:N \premise
4696   \stex_reactivate_macro:N \conclusion
4697   \stex_if_smsmode:F {
4698     \seq_clear:N \l_tmpa_seq
4699     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4700       \tl_if_empty:nF{ ##1 }{
4701         \stex_get_symbol:n { ##1 }
4702         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4703           \l_stex_get_symbol_uri_str
4704         }
4705       }
4706     }
4707     \exp_args:Nnnx
4708     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4709     \str_if_empty:NF \sassertiontype {
4710       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4711     }
4712     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4713 \tl_clear:N \l_tmpa_tl
4714 \clist_map_inline:Nn \l_tmpa_clist {
4715   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4716     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4717   }
4718 }
4719 \tl_if_empty:NTF \l_tmpa_tl {
4720   \__stex_statements_sassertion_start:
4721 }{
4722   \l_tmpa_tl
4723 }
4724 }
4725 \str_if_empty:NTF \sassertionid {
4726   \str_if_empty:NF \sassertionname {
4727     \stex_ref_new_doc_target:n {}
4728   }
4729 } {
4730   \stex_ref_new_doc_target:n \sassertionid
4731 }
4732 \stex_smsmode_do:
4733 ){
4734   \str_if_empty:NF \sassertionname {
4735     \stex_symdecl_do:nn{ }\sassertionname}
4736   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4737 }
4738 \stex_if_smsmode:F {
4739   \clist_set:Nn \l_tmpa_clist \sassertiontype
4740   \tl_clear:N \l_tmpa_tl
4741   \clist_map_inline:Nn \l_tmpa_clist {
4742     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4743       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4744     }
4745   }
4746   \tl_if_empty:NTF \l_tmpa_tl {
4747     \__stex_statements_sassertion_end:
4748   }{
4749     \l_tmpa_tl
4750   }
4751   \end{stex_annotate_env}
4752 }
4753 }

```

\stexpatchassertion

```

4754
4755 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4756   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4757     (\sassertiontitle)
4758   }~}
4759 }
4760 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4761
4762 \newcommand\stexpatchassertion[3] [] {
4763   \str_set:Nx \l_tmpa_str{ #1 }
4764   \str_if_empty:NTF \l_tmpa_str {

```

```

4765     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4766     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4767   }{
4768     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4769     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4770   }
4771 }

```

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

\inlineass inline:

```

4772 \keys_define:nn {stex / inlineass }{
4773   type      .str_set_x:N = \sassertiontype,
4774   id        .str_set_x:N = \sassertionid,
4775   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4776   name      .str_set_x:N = \sassertionname
4777 }
4778 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4779   \str_clear:N \sassertiontype
4780   \str_clear:N \sassertionid
4781   \str_clear:N \sassertionname
4782   \clist_clear:N \l__stex_statements_sassertion_for_clist
4783   \keys_set:nn { stex / inlineass }{ #1 }
4784 }
4785 \NewDocumentCommand \inlineass { 0{} m } {
4786   \beginngroup
4787   \stex_reactivate_macro:N \premise
4788   \stex_reactivate_macro:N \conclusion
4789   \__stex_statements_inlineass_args:n{ #1 }
4790   \str_if_empty:NTF \sassertionid {
4791     \str_if_empty:NF \sassertionname {
4792       \stex_ref_new_doc_target:n {}
4793     }
4794   } {
4795     \stex_ref_new_doc_target:n \sassertionid
4796   }
4797
4798   \stex_if_smsmode:TF{
4799     \str_if_empty:NF \sassertionname {
4800       \stex_symdecl_do:nn{}{\sassertionname}
4801       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4802     }
4803   }{
4804     \seq_clear:N \l_tmpa_seq
4805     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4806       \tl_if_empty:nF{ ##1 }{
4807         \stex_get_symbol:n { ##1 }
4808         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4809           \l_stex_get_symbol_uri_str
4810         }
4811       }
4812     }
4813     \exp_args:Nnx
4814     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4815     \str_if_empty:NF \sassertiontype {
4816       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4817     }
4818     #2
4819     \str_if_empty:NF \sassertionname {
4820       \stex_symdecl_do:nn{}{\sassertionname}
4821       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4822     }
4823   }
4824 }
4825 \endgroup
4826 \stex_smsmode_do:
4827 }

```

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

33.3 Examples

`sexample`

```

4828
4829 \keys_define:nn {stex / sexample }{
4830   type      .str_set_x:N = \exampletype,
4831   id        .str_set_x:N = \sexampleid,
4832   title     .tl_set:N    = \sexampletitle,
4833   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4834 }
4835 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4836   \str_clear:N \sexampletype
4837   \str_clear:N \sexampleid
4838   \tl_clear:N \sexampletitle
4839   \clist_clear:N \l__stex_statements_sexample_for_clist
4840   \keys_set:nn { stex / sexample }{ #1 }
4841 }
4842
4843 \NewDocumentEnvironment{sexample}{0{}}{
4844   \__stex_statements_sexample_args:n{ #1 }
4845   \stex_reactivate_macro:N \premise
4846   \stex_reactivate_macro:N \conclusion
4847   \stex_if_smsmode:F {
4848     \seq_clear:N \l_tmpa_seq
4849     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4850       \tl_if_empty:nF{ ##1 }{
4851         \stex_get_symbol:n { ##1 }
4852         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4853           \l_stex_get_symbol_uri_str
4854         }
4855       }
4856     }
4857     \exp_args:Nnnx
4858     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4859     \str_if_empty:NF \sexampletype {
4860       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4861     }

```



```

4862 \clist_set:No \l_tmpa_clist \sexamplotype
4863 \tl_clear:N \l_tmpa_tl
4864 \clist_map_inline:Nn \l_tmpa_clist {
4865   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4866     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4867   }
4868 }
4869 \tl_if_empty:NTF \l_tmpa_tl {
4870   \__stex_statements_sexample_start:
4871 }{
4872   \l_tmpa_tl
4873 }
4874 }
4875 \str_if_empty:NF \sexampleid {
4876   \stex_ref_new_doc_target:n \sexampleid
4877 }
4878 \stex_smsmode_do:
4879 }{
4880   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4881   \stex_if_smsmode:F {
4882     \clist_set:No \l_tmpa_clist \sexamplotype
4883     \tl_clear:N \l_tmpa_tl
4884     \clist_map_inline:Nn \l_tmpa_clist {
4885       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4886         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4887       }
4888     }
4889     \tl_if_empty:NTF \l_tmpa_tl {
4890       \__stex_statements_sexample_end:
4891     }{
4892       \l_tmpa_tl
4893     }
4894     \end{stex_annotate_env}
4895   }
4896 }

```

\stexpatchexample

```

4897
4898 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4899   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4900     (\sexamplename)
4901   }~}
4902 }
4903 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4904
4905 \newcommand\stexpatchexample[3]{} {
4906   \str_set:Nx \l_tmpa_str{ #1 }
4907   \str_if_empty:NTF \l_tmpa_str {
4908     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4909     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4910   }{
4911     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4912     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4913   }

```

4914 }

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

`\inlineex` inline:

```

4915 \keys_define:nn {stex / inlineex }{
4916   type      .str_set_x:N = \sexamplotype,
4917   id        .str_set_x:N = \sexampleid,
4918   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4919   name      .str_set_x:N = \sexamplename
4920 }
4921 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4922   \str_clear:N \sexamplotype
4923   \str_clear:N \sexampleid
4924   \str_clear:N \sexamplename
4925   \clist_clear:N \l__stex_statements_sexample_for_clist
4926   \keys_set:nn { stex / inlineex }{ #1 }
4927 }
4928 \NewDocumentCommand \inlineex { 0{ } m } {
4929   \begingroup
4930   \stex_reactivate_macro:N \premise
4931   \stex_reactivate_macro:N \conclusion
4932   \__stex_statements_inlineex_args:n{ #1 }
4933   \str_if_empty:NF \sexampleid {
4934     \stex_ref_new_doc_target:n \sexampleid
4935   }
4936   \stex_if_smsmode:TF{
4937     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4938   }{
4939     \seq_clear:N \l_tmpa_seq
4940     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4941       \tl_if_empty:nF{ ##1 }{
4942         \stex_get_symbol:n { ##1 }
4943         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4944           \l_stex_get_symbol_uri_str
4945         }
4946       }
4947     }
4948     \exp_args:Nnx
4949     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4950       \str_if_empty:NF \sexamplotype {
4951         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4952       }
4953       #2
4954       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4955     }
4956   }
4957   \endgroup
4958   \stex_smsmode_do:
4959 }

```

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

33.4 Logical Paragraphs

sparagraph

```

4960 \keys_define:nn { stex / sparagraph } {
4961   id       .str_set_x:N = \sparagraphid ,
4962   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4963   type     .str_set_x:N = \sparagraphtype ,
4964   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4965   from     .tl_set:N    = \sparagraphfrom ,
4966   to       .tl_set:N    = \sparagraphto ,
4967   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4968   name     .str_set:N   = \sparagraphname
4969 }
4970
4971 \cs_new_protected:Nn \stex_sparagraph_args:n {
4972   \tl_clear:N \l_stex_sparagraph_title_tl
4973   \tl_clear:N \sparagraphfrom
4974   \tl_clear:N \sparagraphto
4975   \tl_clear:N \l_stex_sparagraph_start_tl
4976   \str_clear:N \sparagraphid
4977   \str_clear:N \sparagraphtype
4978   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4979   \str_clear:N \sparagraphname
4980   \keys_set:nn { stex / sparagraph } { #1 }
4981 }
4982 \newif\if@in@omtext\@in@omtextfalse
4983
4984 \NewDocumentEnvironment {sparagraph} { 0{ } } {
4985   \stex_sparagraph_args:n { #1 }
4986   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4987     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4988   }{
4989     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4990   }
4991   \@in@omtexttrue
4992   \stex_if_smsmode:F {
4993     \seq_clear:N \l_tmpa_seq
4994     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4995       \tl_if_empty:NF{ ##1 }{
4996         \stex_get_symbol:n { ##1 }
4997         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4998           \l_stex_get_symbol_uri_str
4999         }
5000       }
5001     }
5002     \exp_args:Nnnx
5003     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5004     \str_if_empty:NF \sparagraphtype {
5005       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
5006     }
5007     \str_if_empty:NF \sparagraphfrom {
5008       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5009     }
5010     \str_if_empty:NF \sparagraphto {

```

```

5011     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5012   }
5013   \clist_set:No \l_tmpa_clist \sparagraphtype
5014   \tl_clear:N \l_tmpa_tl
5015   \clist_map_inline:Nn \sparagraphtype {
5016     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5017       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5018     }
5019   }
5020   \tl_if_empty:NTF \l_tmpa_tl {
5021     \__stex_statements_sparagraph_start:
5022   }{
5023     \l_tmpa_tl
5024   }
5025 }
5026 \clist_set:No \l_tmpa_clist \sparagraphtype
5027 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5028   {
5029     \stex_reactivate_macro:N \definiendum
5030     \stex_reactivate_macro:N \definame
5031     \stex_reactivate_macro:N \Definame
5032     \stex_reactivate_macro:N \premise
5033     \stex_reactivate_macro:N \definiens
5034   }
5035   \str_if_empty:NTF \sparagraphid {
5036     \str_if_empty:NTF \sparagraphname {
5037       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5038         \stex_ref_new_doc_target:n {}
5039       }
5040     } {
5041       \stex_ref_new_doc_target:n {}
5042     }
5043   } {
5044     \stex_ref_new_doc_target:n \sparagraphid
5045   }
5046   \exp_args:NNx
5047   \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5048     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5049       \tl_if_empty:nF{ ##1 }{
5050         \stex_get_symbol:n { ##1 }
5051         \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5052       }
5053     }
5054   }
5055   \stex_smsmode_do:
5056   \ignorespacesandpars
5057 }{
5058   \str_if_empty:NF \sparagraphname {
5059     \stex_symdecl_do:nn{}{\sparagraphname}
5060     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5061   }
5062   \stex_if_smsmode:F {
5063     \clist_set:No \l_tmpa_clist \sparagraphtype
5064     \tl_clear:N \l_tmpa_tl

```

```

5065 \clist_map_inline:Nn \l_tmpa_clist {
5066   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5067     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5068   }
5069 }
5070 \tl_if_empty:NTF \l_tmpa_tl {
5071   \__stex_statements_sparagraph_end:
5072 }{
5073   \l_tmpa_tl
5074 }
5075 \end{stex_annotate_env}
5076 }
5077 }

```

\stexpatchparagraph

```

5078
5079 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5080   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5081     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5082       \titleemph{\l_stex_sparagraph_title_tl}:~
5083     }
5084   }{
5085     \titleemph{\l_stex_sparagraph_start_tl}~
5086   }
5087 }
5088 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5089
5090 \newcommand\stexpatchparagraph[3]{} {
5091   \str_set:Nx \l_tmpa_str{ #1 }
5092   \str_if_empty:NTF \l_tmpa_str {
5093     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5094     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5095   }{
5096     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5097     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5098   }
5099 }
5100
5101 \keys_define:nn { stex / inlinepara } {
5102   id      .str_set_x:N = \sparagraphid ,
5103   type    .str_set_x:N = \sparagraphtype ,
5104   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5105   from    .tl_set:N    = \sparagraphfrom ,
5106   to      .tl_set:N    = \sparagraphto ,
5107   name    .str_set:N    = \sparagraphname
5108 }
5109 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5110   \tl_clear:N \sparagraphfrom
5111   \tl_clear:N \sparagraphto
5112   \str_clear:N \sparagraphid
5113   \str_clear:N \sparagraphtype
5114   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5115   \str_clear:N \sparagraphname
5116   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

5117 }
5118 \NewDocumentCommand \inlinepara { 0{} m } {
5119   \beginngroup
5120   \__stex_statements_inlinepara_args:n{ #1 }
5121   \clist_set:No \l_tmpa_clist \sparagraphtype
5122   \str_if_empty:NTF \sparaagraphid {
5123     \str_if_empty:NTF \sparaagraphname {
5124       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5125         \stex_ref_new_doc_target:n {}
5126       }
5127     } {
5128       \stex_ref_new_doc_target:n {}
5129     }
5130   } {
5131     \stex_ref_new_doc_target:n \sparaagraphid
5132   }
5133   \stex_if_smsmode:TF{
5134     \str_if_empty:NF \sparaagraphname {
5135       \stex_symdecl_do:nn{}{\sparaagraphname}
5136       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5137     }
5138   }{
5139     \seq_clear:N \l_tmpa_seq
5140     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
5141       \tl_if_empty:nF{ ##1 }{
5142         \stex_get_symbol:n { ##1 }
5143         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5144           \l_stex_get_symbol_uri_str
5145         }
5146       }
5147     }
5148     \exp_args:Nnx
5149     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5150       \str_if_empty:NF \sparaagraphtype {
5151         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
5152       }
5153       \str_if_empty:NF \sparaagraphfrom {
5154         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5155       }
5156       \str_if_empty:NF \sparaagraphto {
5157         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5158       }
5159       \str_if_empty:NF \sparaagraphname {
5160         \stex_symdecl_do:nn{}{\sparaagraphname}
5161         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5162       }
5163       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5164         \clist_map_inline:Nn \l_tmpa_seq {
5165           \stex_ref_new_sym_target:n {##1}
5166         }
5167       }
5168     } #2
5169   }
5170 }

```

```

5171 \endgroup
5172 \stex_smsmode_do:
5173 }
5174
(End definition for \stexpatchparagraph. This function is documented on page ??.)
5175 </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

```
5176 <*package>
5177 <@@=stex_sproof>
5178
5179 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5180
```

34.2 Proofs

We first define some keys for the proof environment.

```
5181 \keys_define:nn { stex / spf } {
5182   id          .str_set_x:N = \spfid,
5183   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5184   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5185   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5186   type        .str_set_x:N = \spftype,
5187   title       .tl_set:N    = \spftitle,
5188   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5189   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5190   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5191 }
5192 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5193   \str_clear:N \spfid
5194   \tl_clear:N \l__stex_sproof_spf_for_tl
5195   \tl_clear:N \l__stex_sproof_spf_from_tl
5196   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5197   \str_clear:N \spftype
5198   \tl_clear:N \spftitle
5199   \tl_clear:N \l__stex_sproof_spf_continues_tl
5200   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML


```

5201 \tl_clear:N \l__stex_sproof_spf_method_tl
5202 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5203 \keys_set:nn { stex / spf }{ #1 }
5204 }

```

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

```

5205 \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.

```

5206 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5207 \cs_new_protected:Npn \sproofnumber {
5208   \int_set:Nn \l_tmpa_int {1}
5209   \bool_while_do:nn {
5210     \int_compare_p:nNn {
5211       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5212     } > 0
5213   }{
5214     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5215     \int_incr:N \l_tmpa_int
5216   }
5217 }
5218 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5219   \int_set:Nn \l_tmpa_int {1}
5220   \bool_while_do:nn {
5221     \int_compare_p:nNn {
5222       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5223     } > 0
5224   }{
5225     \int_incr:N \l_tmpa_int
5226   }
5227   \int_compare:nNnF \l_tmpa_int = 1 {
5228     \int_decr:N \l_tmpa_int
5229   }
5230   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5231     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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


```

5276     }
5277     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5278       \input{sproof-finnish.ldf}
5279     }
5280     \clist_if_in:NnT \l_tmpa_clist {french}{
5281       \input{sproof-french.ldf}
5282     }
5283     \clist_if_in:NnT \l_tmpa_clist {russian}{
5284       \input{sproof-russian.ldf}
5285     }
5286     \makeatother
5287   }{}
5288 }

```

spfsketch

```

5289 \newcommand\spfsketch[2] [] {
5290   \beginingroup
5291   \let \premise \stex_proof_premise:
5292   \__stex_sproof_spf_args:n{#1}
5293   \stex_if_smsmode:TF {
5294     \str_if_empty:NF \spfid {
5295       \stex_ref_new_doc_target:n \spfid
5296     }
5297   }{
5298     \seq_clear:N \l_tmpa_seq
5299     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5300       \tl_if_empty:nF{ ##1 }{
5301         \stex_get_symbol:n { ##1 }
5302         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5303           \l_stex_get_symbol_uri_str
5304         }
5305       }
5306     }
5307     \exp_args:Nnx
5308     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5309       \str_if_empty:NF \spftype {
5310         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5311       }
5312       \clist_set:Nn \l_tmpa_clist \spftype
5313       \tl_set:Nn \l_tmpa_tl {
5314         \titleemph{
5315           \tl_if_empty:NTF \spftitle {
5316             \spf@proofsketch@kw
5317           }{
5318             \spftitle
5319           }
5320         }:-
5321       }
5322       \clist_map_inline:Nn \l_tmpa_clist {
5323         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5324           \tl_clear:N \l_tmpa_tl
5325         }
5326       }
5327       \str_if_empty:NF \spfid {

```

```

5328         \stex_ref_new_doc_target:n \spfid
5329     }
5330     \l_tmpa_tl #2 \sproofend
5331 }
5332 }
5333 \endgroup
5334 \stex_smsmode_do:
5335 }
5336

```

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

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

```

5337 \newenvironment{spfeq}[2][]{
5338   \__stex_sproof_spf_args:n{#1}
5339   \let \premise \stex_proof_premise:
5340   \stex_if_smsmode:TF {
5341     \str_if_empty:NF \spfid {
5342       \stex_ref_new_doc_target:n \spfid
5343     }
5344   }{
5345     \seq_clear:N \l_tmpa_seq
5346     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5347       \tl_if_empty:NF{ ##1 }{
5348         \stex_get_symbol:n { ##1 }
5349         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5350           \l_stex_get_symbol_uri_str
5351         }
5352       }
5353     }
5354     \exp_args:Nnnx
5355     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5356     \str_if_empty:NF \spftype {
5357       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5358     }
5359
5360     \clist_set:No \l_tmpa_clist \spftype
5361     \tl_clear:N \l_tmpa_tl
5362     \clist_map_inline:Nn \l_tmpa_clist {
5363       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5364         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5365       }
5366       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5367         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5368       }
5369     }
5370     \tl_if_empty:NTF \l_tmpa_tl {
5371       \__stex_sproof_spfeq_start:
5372     }{
5373       \l_tmpa_tl
5374     }{-#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

```

5375 \str_if_empty:NF \spfid {
5376 \stex_ref_new_doc_target:n \spfid
5377 }
5378 \begin{displaymath}\begin{array}{rc1l}
5379 }
5380 \stex_smsmode_do:
5381 }{
5382 \stex_if_smsmode:F {
5383 \end{array}\end{displaymath}
5384 \clist_set:No \l_tmpa_clist \spftype
5385 \tl_clear:N \l_tmpa_tl
5386 \clist_map_inline:Nn \l_tmpa_clist {
5387 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5388 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5389 }
5390 }
5391 \tl_if_empty:NTF \l_tmpa_tl {
5392 \__stex_sproof_spfeq_end:
5393 }{
5394 \l_tmpa_tl
5395 }
5396 \end{stex_annotate_env}
5397 }
5398 }
5399
5400 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5401 \titleemph{
5402 \tl_if_empty:NTF \spftitle {
5403 \spf@proof@kw
5404 }{
5405 \spftitle
5406 }
5407 }:
5408 }
5409 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5410
5411 \newcommand\stexpatchspfeq[3] [] {
5412 \str_set:Nx \l_tmpa_str{ #1 }
5413 \str_if_empty:NTF \l_tmpa_str {
5414 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5415 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5416 }{
5417 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5418 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5419 }
5420 }
5421

```

(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.

```

5422 \newenvironment{sproof}[2] []{

```

```

5423 \let \premise \stex_proof_premise:
5424 \intarray_gzero:N \l__stex_sproof_counter_intarray
5425 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5426 \__stex_sproof_spf_args:n{#1}
5427 \stex_if_smsmode:TF {
5428   \str_if_empty:NF \spfid {
5429     \stex_ref_new_doc_target:n \spfid
5430   }
5431 }{
5432   \seq_clear:N \l_tmpa_seq
5433   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5434     \tl_if_empty:NF{ ##1 }{
5435       \stex_get_symbol:n { ##1 }
5436       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5437         \l_stex_get_symbol_uri_str
5438       }
5439     }
5440   }
5441   \exp_args:Nnnx
5442   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5443   \str_if_empty:NF \spftype {
5444     \stex_annotate_invisible:nnn{type}{\spftype}{}
5445   }
5446
5447   \clist_set:No \l_tmpa_clist \spftype
5448   \tl_clear:N \l_tmpa_tl
5449   \clist_map_inline:Nn \l_tmpa_clist {
5450     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5451       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5452     }
5453     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5454       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5455     }
5456   }
5457   \tl_if_empty:NTF \l_tmpa_tl {
5458     \__stex_sproof_sproof_start:
5459   }{
5460     \l_tmpa_tl
5461   }{~#2}
5462   \str_if_empty:NF \spfid {
5463     \stex_ref_new_doc_target:n \spfid
5464   }
5465   \begin{description}
5466 }{
5467   \stex_smsmode_do:
5468 }{
5469   \stex_if_smsmode:F{
5470     \end{description}
5471     \clist_set:No \l_tmpa_clist \spftype
5472     \tl_clear:N \l_tmpa_tl
5473     \clist_map_inline:Nn \l_tmpa_clist {
5474       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5475         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5476       }

```

```

5477     }
5478     \tl_if_empty:NTF \l_tmpa_tl {
5479       \__stex_sproof_sproof_end:
5480     }{
5481       \l_tmpa_tl
5482     }
5483     \end{stex_annotate_env}
5484   }
5485 }
5486
5487 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5488   \par\noindent\titleemph{
5489     \tl_if_empty:NTF \spftype {
5490       \spf@proof@kw
5491     }{
5492       \spftype
5493     }
5494   }:
5495 }
5496 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5497
5498 \newcommand\stexpatchsproof[3] [] {
5499   \str_set:Nx \l_tmpa_str{ #1 }
5500   \str_if_empty:NTF \l_tmpa_str {
5501     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5502     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5503   }{
5504     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5505     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5506   }
5507 }

```

\spfidea

```

5508 \newcommand\spfidea[2] []{
5509   \__stex_sproof_spf_args:n{#1}
5510   \titleemph{
5511     \tl_if_empty:NTF \spftype {Proof~Idea}{
5512       \spftype
5513     }:
5514   }~#2
5515   \sproofend
5516 }

```

(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

```

5517 \newenvironment{spfstep}[1] []{
5518   \__stex_sproof_spf_args:n{#1}
5519   \stex_if_smsmode:TF {

```

```

5520 \str_if_empty:NF \spfid {
5521   \stex_ref_new_doc_target:n \spfid
5522 }
5523 }{
5524   \@in@omtexttrue
5525   \seq_clear:N \l_tmpa_seq
5526   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5527     \tl_if_empty:NF{ ##1 }{
5528       \stex_get_symbol:n { ##1 }
5529       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5530         \l_stex_get_symbol_uri_str
5531       }
5532     }
5533   }
5534   \exp_args:Nnnx
5535   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5536   \str_if_empty:NF \spftype {
5537     \stex_annotate_invisible:nnn{type}{\spftype}{}
5538   }
5539   \clist_set:No \l_tmpa_clist \spftype
5540   \tl_set:Nn \l_tmpa_tl {
5541     \item[\sproofnumber]
5542     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5543   }
5544   \clist_map_inline:Nn \l_tmpa_clist {
5545     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5546       \tl_clear:N \l_tmpa_tl
5547     }
5548   }
5549   \l_tmpa_tl
5550   \tl_if_empty:NF \spftitle {
5551     {(\titleemph{\spftitle})\enspace}
5552   }
5553   \str_if_empty:NF \spfid {
5554     \stex_ref_new_doc_target:n \spfid
5555   }
5556 }
5557 \stex_smsmode_do:
5558 \ignorespacesandpars
5559 }{
5560   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5561     \__stex_sproof_inc_counter:
5562   }
5563   \stex_if_smsmode:F {
5564     \end{stex_annotate_env}
5565   }
5566 }

```

sproofcomment

```

5567 \newenvironment{sproofcomment}[1][]{
5568   \__stex_sproof_spf_args:n{#1}
5569   \clist_set:No \l_tmpa_clist \spftype
5570   \tl_set:Nn \l_tmpa_tl {
5571     \item[\sproofnumber]

```



```

5572 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5573 }
5574 \clist_map_inline:Nn \l_tmpa_clist {
5575   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5576     \tl_clear:N \l_tmpa_tl
5577   }
5578 }
5579 \l_tmpa_tl
5580 }{
5581   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5582     \__stex_sproof_inc_counter:
5583   }
5584 }

```

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.

```

5585 \newenvironment{subproof}[2][]{
5586   \__stex_sproof_spf_args:n{#1}
5587   \stex_if_smsmode:TF{
5588     \str_if_empty:NF \spfid {
5589       \stex_ref_new_doc_target:n \spfid
5590     }
5591   }{
5592     \seq_clear:N \l_tmpa_seq
5593     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5594       \tl_if_empty:nF{ ##1 }{
5595         \stex_get_symbol:n { ##1 }
5596         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5597           \l_stex_get_symbol_uri_str
5598         }
5599       }
5600     }
5601     \exp_args:Nnnx
5602     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5603     \str_if_empty:NF \spftype {
5604       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5605     }
5606
5607     \clist_set:No \l_tmpa_clist \spftype
5608     \tl_set:Nn \l_tmpa_tl {
5609       \item[\sproofnumber]
5610       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5611     }
5612     \clist_map_inline:Nn \l_tmpa_clist {
5613       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5614         \tl_clear:N \l_tmpa_tl
5615       }
5616     }
5617     \l_tmpa_tl
5618     \tl_if_empty:NF \spftitle {
5619       {(\titleemph{\spftitle})\enspace}
5620     }

```

```

5621     {~#2}
5622     \str_if_empty:NF \spfid {
5623       \stex_ref_new_doc_target:n \spfid
5624     }
5625   }
5626   \__stex_sproof_add_counter:
5627   \stex_smsmode_do:
5628 }{
5629   \__stex_sproof_remove_counter:
5630   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5631     \__stex_sproof_inc_counter:
5632   }
5633   \stex_if_smsmode:F{
5634     \end{stex_annotate_env}
5635   }
5636 }

```

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

```

5637 \newenvironment{spfcases}[2][]{
5638   \tl_if_empty:nTF{#1}{
5639     \begin{subproof}[method=by-cases]{#2}
5640   }{
5641     \begin{subproof}[#1,method=by-cases]{#2}
5642   }
5643 }{
5644   \end{subproof}
5645 }

```

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

```

5646 \newenvironment{spfcase}[2][]{
5647   \__stex_sproof_spf_args:n{#1}
5648   \stex_if_smsmode:TF {
5649     \str_if_empty:NF \spfid {
5650       \stex_ref_new_doc_target:n \spfid
5651     }
5652   }{
5653     \seq_clear:N \l_tmpa_seq
5654     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5655       \tl_if_empty:nF{ ##1 }{
5656         \stex_get_symbol:n { ##1 }
5657         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5658           \l_stex_get_symbol_uri_str
5659         }
5660       }
5661     }
5662     \exp_args:Nnnx
5663     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5664     \str_if_empty:NF \spftype {
5665       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5666   }
5667   \clist_set:Nn \l_tmpa_clist \spftype
5668   \tl_set:Nn \l_tmpa_tl {
5669     \item[\sproofnumber]

```

```

5670     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5671   }
5672   \clist_map_inline:Nn \l_tmpa_clist {
5673     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5674       \tl_clear:N \l_tmpa_tl
5675     }
5676   }
5677   \l_tmpa_tl
5678   \tl_if_empty:nF{#2}{
5679     \titleemph{#2}:~
5680   }
5681 }
5682 \__stex_sproof_add_counter:
5683 \stex_smsmode_do:
5684 ){
5685   \__stex_sproof_remove_counter:
5686   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5687     \__stex_sproof_inc_counter:
5688   }
5689   \stex_if_smsmode:F{
5690     \clist_set:No \l_tmpa_clist \spftype
5691     \tl_set:Nn \l_tmpa_tl{\sproofend}
5692     \clist_map_inline:Nn \l_tmpa_clist {
5693       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5694         \tl_clear:N \l_tmpa_tl
5695       }
5696     }
5697     \l_tmpa_tl
5698     \end{stex_annotate_env}
5699   }
5700 }

```

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

```

5701 \newcommand\spfcasesketch[3][]{
5702   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5703 }

```

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.

```

5704 \keys_define:nn { stex / just }{
5705   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5706   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5707   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5708   args     .tl_set:N    = \l__stex_sproof_just_args_tl
5709 }

```

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

```
5710 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5711 \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.

```
5712 \newcommand\justarg[2] [] {#2}
```

```
5713 \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

```
5714 <*package>
5715
5716 %%%%%%%%%%% others.dtx %%%%%%%%%%%
5717
5718 <@@=stex_others>
    Warnings and error messages
5719 % None

\MSC Math subject classifier

5720 \NewDocumentCommand \MSC {m} {
5721 % TODO
5722 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5723 \@ifpackageloaded{tikzinput}{
5724 \RequirePackage{stex-tikzinput}
5725 }{}
5726 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
5727 <*package>
5728 <@@=stex_modules>
5729
5730 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5731
5732 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5733 \begingroup
5734 \stex_module_setup:nn{
5735   ns=\c_stex_metatheory_ns_str,
5736   meta=NONE
5737 }{Metatheory}
5738 \stex_reactivate_macro:N \symdecl
5739 \stex_reactivate_macro:N \notation
5740 \stex_reactivate_macro:N \symdef
5741 \ExplSyntaxOff
5742 \csname stex_suppress_html:n\endcsname{
5743   % is-a (a:A, a \in A, a is an A, etc.)
5744   \symdecl{isa}[args=ai]
5745   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5746   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5747   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5748
5749   % bind (\forall, \Pi, \lambda etc.)
5750   \symdecl{bind}[args=Bi]
5751   \notation{bind}{forall}{\comp\forall #1.;#2}{##1 \comp, ##2}
5752   \notation{bind}{Pi}{\comp\prod_{#1}#2}{##1 \comp, ##2}
5753   \notation{bind}{deffun}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5754
5755   % implicit bind
5756   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5757
5758   % dummy variable
5759   \symdecl{dummyvar}
5760   \notation{dummyvar}[underscore]{\comp\_}
5761   \notation{dummyvar}[dot]{\comp\cdot}
```

```

5762 \notation{dummyvar}[dash]{\comp{\rm --}}
5763
5764 %fromto (function space, Hom-set, implication etc.)
5765 \symdecl{fromto}[args=ai]
5766 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5767 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5768
5769 % mapto (lambda etc.)
5770 \symdecl{mapto}[args=Bi]
5771 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5772 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5773 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5774
5775 % function/operator application
5776 \symdecl{apply}[args=ia]
5777 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5778 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5779
5780 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5781 \symdecl{metacollection}
5782 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5783 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5784
5785 % collection of propositions/booleans/truth values
5786 \symdecl{prop}[name=proposition]
5787 \notation{prop}[prop]{\comp{\rm prop}}
5788 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5789
5790 % sequences
5791 \symdecl{seqtype}[args=1]
5792 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5793
5794 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{1}_{#2}}
5795 \notation{sequence-index}[ui,prec=nobrackets]{#{1}^{\comp\ast}}
5796
5797 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5798 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5799 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5800
5801 % letin (‘‘let’’, local definitions, variable substitution)
5802 \symdecl{letin}[args=bii]
5803 \notation{letin}[let]{\comp{\rm let}};#1\comp{=}#2;\comp{\rm in}}\;#3}
5804 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5805 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5806
5807 % structures
5808 \symdecl*{module-type}[args=1]
5809 \notation{module-type}{\mathtt{MOD} #1}
5810 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5811 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5812
5813 }
5814 \ExplSyntaxOn
5815 \stex_add_to_current_module:n{

```

```

5816 \let\nappa\apply
5817 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5818 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5819 \def\livar{\csname sequence-index\endcsname[li]}
5820 \def\uivar{\csname sequence-index\endcsname[ui]}
5821 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5822 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5823 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5824 }
5825 \__stex_modules_end_module:
5826 \endgroup
5827 </package>

```


Chapter 37

Tikzinput Implementation

```
5828 <*package>
5829
5830 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5831
5832 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5833 \RequirePackage{l3keys2e}
5834
5835 \keys_define:nn { tikzinput } {
5836   image .bool_set:N = \c_tikzinput_image_bool,
5837   image .default:n = false ,
5838   unknown .code:n = {}
5839 }
5840
5841 \ProcessKeysOptions { tikzinput }
5842
5843 \bool_if:NTF \c_tikzinput_image_bool {
5844   \RequirePackage{graphicx}
5845
5846   \providecommand\usetikzlibrary[]{}
5847   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
5848 }{
5849   \RequirePackage{tikz}
5850   \RequirePackage{standalone}
5851
5852   \newcommand \tikzinput [2] [] {
5853     \setkeys{Gin}{#1}
5854     \ifx \Gin@ewidth \Gin@exclamation
5855       \ifx \Gin@eheight \Gin@exclamation
5856         \input { #2 }
5857       \else
5858         \resizebox{!}{ \Gin@eheight }{
5859           \input { #2 }
5860         }
5861       \fi
5862     \else
5863       \ifx \Gin@eheight \Gin@exclamation
5864         \resizebox{ \Gin@ewidth }{!}{
5865           \input { #2 }
```

```

5866     }
5867     \else
5868         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5869             \input { #2 }
5870         }
5871     \fi
5872 \fi
5873 }
5874 }
5875
5876 \newcommand \ctikzinput [2] [] {
5877     \begin{center}
5878         \tikzinput [1] {#2}
5879     \end{center}
5880 }
5881
5882 \@ifpackageloaded{stex}{
5883     \RequirePackage{stex-tikzinput}
5884 }{}
5885
5886 </package>
5887 <*stex>
5888 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5889 \RequirePackage{stex}
5890 \RequirePackage{tikzinput}
5891
5892 \newcommand\mhtikzinput [2] [] {%
5893     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5894     \stex_in_repository:nn\Gin@mhrepos{
5895         \tikzinput [1]{\mhpath{##1}{#2}}
5896     }
5897 }
5898 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5899 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

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

```
5900 \*cls)
5901 \<@@=document_structure>
5902 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5903 \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`

```
5904 \keys_define:nn{ document-structure / pkg }{
5905   class      .str_set_x:N = \c_document_structure_class_str,
5906   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5907   report     .code:n      = {
5908     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5909     \str_set:Nn \c_document_structure_class_str {report}
5910   },
5911   book       .code:n      = {
5912     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5913     \str_set:Nn \c_document_structure_class_str {book}
5914   },
5915   bookpart   .code:n      = {
5916     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5917     \str_set:Nn \c_document_structure_class_str {book}
5918     \str_set:Nn \c_document_structure_topsect_str {chapter}
5919   },
```

```

5920 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5921 unknown     .code:n      = {
5922   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5923 }
5924 }
5925 \ProcessKeysOptions{ document-structure / pkg }
5926 \str_if_empty:NT \c_document_structure_class_str {
5927   \str_set:Nn \c_document_structure_class_str {article}
5928 }
5929 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5930   {\c_document_structure_class_str}
5931

```

38.3 Beefing up the document environment

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

```

5932 \RequirePackage{document-structure}
5933 \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

```

5934 \keys_define:nn { document-structure / document }{
5935   id .str_set_x:N = \c_document_structure_document_id_str
5936 }
5937 \let\__document_structure_orig_document=\document
5938 \renewcommand{\document}[1][]{
5939   \keys_set:nn{ document-structure / document }{ #1 }
5940   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5941   \__document_structure_orig_document
5942 }

```

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

```

5943 }
5944 \</cls>

```

38.4 Implementation: document-structure Package

```

5945 \<*package>
5946 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
5947 \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

```

5948
5949 \keys_define:nn{ document-structure / pkg }{
5950   class      .str_set_x:N = \c_document_structure_class_str,
5951   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5952   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5953 }
5954 \ProcessKeysOptions{ document-structure / pkg }
5955 \str_if_empty:NT \c_document_structure_class_str {
5956   \str_set:Nn \c_document_structure_class_str {article}
5957 }
5958 \str_if_empty:NT \c_document_structure_topsect_str {
5959   \str_set:Nn \c_document_structure_topsect_str {section}
5960 }

```

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

```

5961 \RequirePackage{xspace}
5962 \RequirePackage{comment}
5963 \AddToHook{begindocument}{
5964   \ltx@ifpackageloaded{babel}{
5965     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5966     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5967       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
5968     }
5969   }{}
5970 }

```

`\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.

```

5971 \int_new:N \l_document_structure_section_level_int
5972 \str_case:VnF \c_document_structure_topsect_str {
5973   {part}}{
5974     \int_set:Nn \l_document_structure_section_level_int {0}
5975   }
5976   {chapter}}{
5977     \int_set:Nn \l_document_structure_section_level_int {1}
5978   }
5979 }{
5980   \str_case:VnF \c_document_structure_class_str {
5981     {book}}{
5982       \int_set:Nn \l_document_structure_section_level_int {0}
5983     }
5984     {report}}{
5985       \int_set:Nn \l_document_structure_section_level_int {0}
5986     }
5987   }{
5988     \int_set:Nn \l_document_structure_section_level_int {2}
5989   }
5990 }

```

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

```
5991 \def\current@section@level{document}%
5992 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5993 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5994 \cs_new_protected:Npn \skipomgroup {
5995   \ifcase\l_document_structure_section_level_int
5996   \or\stepcounter{part}
5997   \or\stepcounter{chapter}
5998   \or\stepcounter{section}
5999   \or\stepcounter{subsection}
6000   \or\stepcounter{subsubsection}
6001   \or\stepcounter{paragraph}
6002   \or\stepcounter{subparagraph}
6003   \fi
6004 }
```

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

`blindfragment`

```
6005 \newcommand\at@begin@blindomgroup[1]{%
6006 \newenvironment{blindfragment}
6007 {
6008   \int_incr:N\l_document_structure_section_level_int
6009   \at@begin@blindomgroup\l_document_structure_section_level_int
6010 }{}}
```

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

```
6011 \newcommand\omgroup@nonum[2]{
6012   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6013   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6014 }
```

(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.

```
6015 \newcommand\omgroup@num[2]{
```

¹⁸EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6016 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6017   \@nameuse{#1}{#2}
6018 }{
6019   \cs_if_exist:NTF\rdfmata@sectioning{
6020     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6021   }{
6022     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6023   }
6024 }
6025 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6026 }

```

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

sfragment

```

6027 \keys_define:nn { document-structure / omgroup }{
6028   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6029   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6030   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6031   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6032   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6033   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6034   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6035   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6036   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6037   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6038 }
6039 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6040   \str_clear:N \l__document_structure_omgroup_id_str
6041   \str_clear:N \l__document_structure_omgroup_date_str
6042   \clist_clear:N \l__document_structure_omgroup_creators_clist
6043   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6044   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6045   \tl_clear:N \l__document_structure_omgroup_type_tl
6046   \tl_clear:N \l__document_structure_omgroup_short_tl
6047   \tl_clear:N \l__document_structure_omgroup_display_tl
6048   \tl_clear:N \l__document_structure_omgroup_intro_tl
6049   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6050   \keys_set:nn { document-structure / omgroup } { #1 }
6051 }

```

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.

```

6052 \newif\if@mainmatter\@mainmattertrue
6053 \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.

```

6054 \keys_define:nn { document-structure / sectioning }{
6055   name .str_set_x:N = \l__document_structure_sect_name_str ,
6056   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6057   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6058   clear .default:n = {true} ,
6059   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6060   num      .default:n      = {true}
6061 }
6062 \cs_new_protected:Nn \__document_structure_sect_args:n {
6063   \str_clear:N \l__document_structure_sect_name_str
6064   \str_clear:N \l__document_structure_sect_ref_str
6065   \bool_set_false:N \l__document_structure_sect_clear_bool
6066   \bool_set_false:N \l__document_structure_sect_num_bool
6067   \keys_set:nn { document-structure / sectioning } { #1 }
6068 }
6069 \newcommand\omdoc@sectioning[3][]{
6070   \__document_structure_sect_args:n {#1}
6071   \let\omdoc@sect@name\l__document_structure_sect_name_str
6072   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6073   \if@mainmatter% numbering not overridden by frontmatter, etc.
6074     \bool_if:NTF \l__document_structure_sect_num_bool {
6075       \omgroup@num{#2}{#3}
6076     }{
6077       \omgroup@nonum{#2}{#3}
6078     }
6079     \def\current@section@level{\omdoc@sect@name}
6080   \else
6081     \omgroup@nonum{#2}{#3}
6082   \fi
6083 }% 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.

```

6084 \newcommand\omgroup@redefine@addtocontents[1]{%
6085   %\edef\__document_structureimport{#1}%
6086   %\@for\@I:=\__document_structureimport\do{%
6087     %\edef\@path{\csname module@\@I @path\endcsname}%
6088     %\@ifundefined{tf@toc}\relax%
6089     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6090   %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6091   %\def\addcontentsline##1##2##3{%
6092     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6093   %\else% hyperref.sty not loaded
6094   %\def\addcontentsline##1##2##3{%
6095     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6096   %\fi
6097 }% 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.

```

6098 \newenvironment{sfragment}[2][]{% keys, title
6099 {
6100   \__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.

```

6101   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6102     \omgroup@redefine@addtocontents{
6103       %\@ifundefined{module@id}\used@modules%

```



```

6104     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6105   }
6106 }

```

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

```

6107 \int_incr:N\l_document_structure_section_level_int
6108 \ifcase\l_document_structure_section_level_int
6109   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6110   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6111   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6112   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6113   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6114   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6115   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6116 \fi
6117 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6118 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6119   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6120 }
6121 }% for customization
6122 {}

```

and finally, we localize the sections

```

6123 \newcommand\omdoc@part@kw{Part}
6124 \newcommand\omdoc@chapter@kw{Chapter}
6125 \newcommand\omdoc@section@kw{Section}
6126 \newcommand\omdoc@subsection@kw{Subsection}
6127 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6128 \newcommand\omdoc@paragraph@kw{paragraph}
6129 \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`

```

6130 \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).

```

6131 \cs_if_exist:NTF\frontmatter{
6132   \let\__document_structure_orig_frontmatter\frontmatter
6133   \let\frontmatter\relax
6134 }{
6135   \tl_set:Nn\__document_structure_orig_frontmatter{
6136     \clearpage
6137     \@mainmatterfalse
6138     \pagenumbering{roman}

```

```

6139 }
6140 }
6141 \cs_if_exist:NTF\backmatter{
6142   \let\__document_structure_orig_backmatter\backmatter
6143   \let\backmatter\relax
6144 }{
6145   \tl_set:Nn\__document_structure_orig_backmatter{
6146     \clearpage
6147     \@mainmatterfalse
6148     \pagenumbering{roman}
6149   }
6150 }

```

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.

```

6151 \newenvironment{frontmatter}{
6152   \__document_structure_orig_frontmatter
6153 }{
6154   \cs_if_exist:NTF\mainmatter{
6155     \mainmatter
6156   }{
6157     \clearpage
6158     \@mainmattertrue
6159     \pagenumbering{arabic}
6160   }
6161 }

```

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

```

6162 \newenvironment{backmatter}{
6163   \__document_structure_orig_backmatter
6164 }{
6165   \cs_if_exist:NTF\mainmatter{
6166     \mainmatter
6167   }{
6168     \clearpage
6169     \@mainmattertrue
6170     \pagenumbering{arabic}
6171   }
6172 }

```

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

```

6173 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6174 \def \c__document_structure_document_str{document}
6175 \newcommand\afterprematurestop{}
6176 \def\prematurestop@endomgroup{
6177   \unless\ifx\@currenvir\c__document_structure_document_str
6178     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
6179     \expandafter\prematurestop@endomgroup

```

```

6180 \fi
6181 }
6182 \providecommand\prematurestop{
6183 \message{Stopping~sTeX~processing~prematurely}
6184 \prematurestop@endomgroup
6185 \afterprematurestop
6186 \end{document}
6187 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

6188 \RequirePackage{etoolbox}
6189 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6190 \newrobustcmd\useSGvar[1]{%
6191 \@ifundefined{sTeX@Gvar@#1}
6192 {\PackageError{document-structure}
6193 {The sTeX Global variable #1 is undefined}
6194 {set it with \protect\setSGvar}}
6195 \@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.

```

6196 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6197 \@ifundefined{sTeX@Gvar@#1}
6198 {\PackageError{document-structure}
6199 {The sTeX Global variable #1 is undefined}
6200 {set it with \protect\setSGvar}}
6201 {\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.

```
6202 \*cls)
6203 \@@=notesslides)
6204 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6205 \RequirePackage{13keys2e}
6206
6207 \keys_define:nn{notesslides / cls}{
6208   class .code:n = {
6209     \PassOptionsToClass{\CurrentOption}{document-structure}
6210     \str_if_eq:nnT{#1}{book}{
6211       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6212     }
6213     \str_if_eq:nnT{#1}{report}{
6214       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6215     }
6216   },
6217   notes .bool_set:N = \c__notesslides_notes_bool ,
6218   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6219   unknown .code:n = {
6220     \PassOptionsToClass{\CurrentOption}{document-structure}
6221     \PassOptionsToClass{\CurrentOption}{beamer}
6222     \PassOptionsToPackage{\CurrentOption}{notesslides}
6223   }
6224 }
6225 \ProcessKeysOptions{ notesslides / cls }
6226 \bool_if:NTF \c__notesslides_notes_bool {
6227   \PassOptionsToPackage{notes=true}{notesslides}
6228 }{
6229   \PassOptionsToPackage{notes=false}{notesslides}
6230 }
6231 \</cls)
```

now we do the same for the notesslides package.

```

6232 <*package>
6233 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6234 \RequirePackage{13keys2e}
6235
6236 \keys_define:nn{notesslides / pkg}{
6237   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6238   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6239   notes        .bool_set:N = \c_notesslides_notes_bool ,
6240   slides        .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6241   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6242   frameimages   .bool_set:N = \c_notesslides_frameimages_bool ,
6243   fiboxed        .bool_set:N = \c_notesslides_fiboxed_bool ,
6244   nopproblems    .bool_set:N = \c_notesslides_nopproblems_bool,
6245   unknown        .code:n      = {
6246     \PassOptionsToClass{\CurrentOption}{stex}
6247     \PassOptionsToClass{\CurrentOption}{tikzinput}
6248   }
6249 }
6250 \ProcessKeysOptions{ notesslides / pkg }
6251 \newif\ifnotes
6252 \bool_if:NTF \c_notesslides_notes_bool {
6253   \notesttrue
6254 }{
6255   \notesfalse
6256 }
6257

```

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

```

6258 \str_if_empty:NTF \c_notesslides_topsect_str {
6259   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
6260 }{
6261   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
6262 }
6263 </package>

```

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

```

6264 <*cls>
6265 \bool_if:NTF \c_notesslides_notes_bool {
6266   \LoadClass{document-structure}
6267 }{
6268   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6269   \newcounter{Item}
6270   \newcounter{paragraph}
6271   \newcounter{subparagraph}
6272   \newcounter{Hfootnote}
6273   \RequirePackage{document-structure}
6274 }

```

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

```

6275 \RequirePackage{notesslides}
6276 </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.

```

6277 \*package>
6278 \bool_if:NT \c__notesslides_notes_bool {
6279   \RequirePackage{a4wide}
6280   \RequirePackage{marginnote}
6281   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6282   \RequirePackage{mdframed}
6283   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6284   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6285 }
6286 \RequirePackage{stex-tikzinput}
6287 \RequirePackage{etoolbox}
6288 \RequirePackage{amssymb}
6289 \RequirePackage{amsmath}
6290 \RequirePackage{comment}
6291 \RequirePackage{textcomp}
6292 \RequirePackage{url}
6293 \RequirePackage{graphicx}
6294 \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

```

6295 \bool_if:NT \c__notesslides_notes_bool {
6296   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6297 }
6298
6299
6300 \NewDocumentCommand \libusetheme {0{} m} {
6301   \bool_if:NTF \c__notesslides_notes_bool {
6302     \libusepackage[#1]{beamernotestheme#2}
6303   }{
6304     \libusepackage[#1]{beamertheme#2}
6305   }
6306 }

```

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

```

6307 \newcounter{slide}
6308 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6309 \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.

```

6310 \bool_if:NTF \c__notesslides_notes_bool {
6311   \renewenvironment{note}{\ignorespaces}{}
6312 }{
6313   \excludecomment{note}
6314 }

```

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.

```

6315 \bool_if:NT \c__notesslides_notes_bool {
6316   \newlength{\slideframewidth}
6317   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6318 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6319   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6320     \bool_set_true:N #1
6321   }{
6322     \bool_set_false:N #1
6323   }
6324 }
6325 \keys_define:nn{notesslides / frame}{
6326   label .str_set_x:N = \l__notesslides_frame_label_str,
6327   allowframebreaks .code:n = {
6328     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6329   },
6330   allowdisplaybreaks .code:n = {
6331     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6332   },
6333   fragile .code:n = {
6334     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6335   },
6336   shrink .code:n = {
6337     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6338   },
6339   squeeze .code:n = {
6340     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6341   },
6342   t .code:n = {
6343     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6344   },
6345 }
6346 \cs_new_protected:Nn \__notesslides_frame_args:n {
6347   \str_clear:N \l__notesslides_frame_label_str
6348   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6349   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6350   \bool_set_true:N \l__notesslides_frame_fragile_bool
6351   \bool_set_true:N \l__notesslides_frame_shrink_bool
6352   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6353   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6354 \keys_set:nn { notesslides / frame }{ #1 }
6355 }

```

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

```

6356 \renewenvironment{frame}[1][]{
6357   \__notesslides_frame_args:n{#1}
6358   \sffamily
6359   \stepcounter{slide}
6360   \def\@currentlabel{\theslide}
6361   \str_if_empty:NF \l__notesslides_frame_label_str {
6362     \label{\l__notesslides_frame_label_str}
6363   }

```

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

```

6364 \def\itemize@level{outer}
6365 \def\itemize@outer{outer}
6366 \def\itemize@inner{inner}
6367 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6368 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6369 \renewenvironment{itemize}{
6370   \ifx\itemize@level\itemize@outer
6371     \def\itemize@label{$\rhd$}
6372   \fi
6373   \ifx\itemize@level\itemize@inner
6374     \def\itemize@label{$\scriptstyle\rhd$}
6375   \fi
6376   \begin{list}
6377     {\itemize@label}
6378     {\setlength{\labelsep}{.3em}
6379      \setlength{\labelwidth}{.5em}
6380      \setlength{\leftmargin}{1.5em}
6381     }
6382   \edef\itemize@level{\itemize@inner}
6383 }{
6384   \end{list}
6385 }

```

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

```

6386 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6387 }{
6388   \medskip\miko@slidelabel\end{mdframed}
6389 }

```

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

`\frametitle`

```

6390 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6391 }

```

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

EdN:20

`\pause` 20

```

6392 \bool_if:NT \c__notesslides_notes_bool {
6393   \newcommand\pause{}
6394 }

```

²⁰EdNOTE: MK: fake it in notes mode for now

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

nparagraph

```
6395 \bool_if:NTF \c__notesslides_notes_bool {
6396   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6397 }{
6398   \excludecomment{nparagraph}
6399 }
```

nfragment

```
6400 \bool_if:NTF \c__notesslides_notes_bool {
6401   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6402 }{
6403   \excludecomment{nfragment}
6404 }
```

ndefinition

```
6405 \bool_if:NTF \c__notesslides_notes_bool {
6406   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6407 }{
6408   \excludecomment{ndefinition}
6409 }
```

nassertion

```
6410 \bool_if:NTF \c__notesslides_notes_bool {
6411   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6412 }{
6413   \excludecomment{nassertion}
6414 }
```

nsproof

```
6415 \bool_if:NTF \c__notesslides_notes_bool {
6416   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6417 }{
6418   \excludecomment{nsproof}
6419 }
```

nexample

```
6420 \bool_if:NTF \c__notesslides_notes_bool {
6421   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6422 }{
6423   \excludecomment{nexample}
6424 }
```

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

```
6425 \def\inputref@preskip{\smallskip}
6426 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6427 \let\orig@inputref\inputref
6428 \def\inputref{\@ifstar\ninputref\orig@inputref}
6429 \newcommand\ninputref[2][]{
6430   \bool_if:NT \c__notesslides_notes_bool {
6431     \orig@inputref[#1]{#2}
6432   }
6433 }
```

(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>}`.

```
6434 \newlength{\slidelogoheight}
6435
6436 \bool_if:NTF \c__notesslides_notes_bool {
6437   \setlength{\slidelogoheight}{.4cm}
6438 }{
6439   \setlength{\slidelogoheight}{1cm}
6440 }
6441 \newsavebox{\slidelogo}
6442 \sbox{\slidelogo}{\sTeX}
6443 \newrobustcmd{\setslidelogo}[1]{\def\source{#1}}
6444 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6445 }
```

(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.

```
6446 \def\source{Michael Kohlhase}% customize locally
6447 \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.

```
6448 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6449 \newsavebox{\cclogo}
6450 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6451 \newif\ifcchref\cchreffalse
6452 \AtBeginDocument{
6453   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6454 }
6455 \def\licensing{
6456   \ifcchref
```

```

6457 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6458 \else
6459 {\usebox{\cclogo}}
6460 \fi
6461 }
6462 \newrobustcmd{\setlicensing}[2][]{
6463 \def\@url{#1}
6464 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6465 \ifx\@url\@empty
6466 \def\licensing{\usebox{\cclogo}}
6467 \else
6468 \def\licensing{
6469 \ifcchref
6470 \href{#1}{\usebox{\cclogo}}
6471 \else
6472 {\usebox{\cclogo}}
6473 \fi
6474 }
6475 \fi
6476 }

```

(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

```

6477 \newrobustcmd\miko@slidelabel{
6478 \vbox to \slidelogoheight{
6479 \vss\hbox to \slidewidth
6480 {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6481 }
6482 }

```

(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.

```

6483 \def\Gin@mhrepos{}
6484 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6485 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6486 \newrobustcmd\frameimage[2][]{
6487 \stepcounter{slide}
6488 \bool_if:NT \c__notesslides_frameimages_bool {
6489 \def\Gin@ewidth{}\setkeys{Gin}{#1}
6490 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6491 \begin{center}
6492 \bool_if:NTF \c__notesslides_fiboxed_bool {
6493 \fbox{
6494 \ifx\Gin@ewidth\@empty
6495 \ifx\Gin@mhrepos\@empty
6496 \mhgraphics[width=\slidewidth,#1]{#2}
6497 \else

```

²¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6498         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6499     \fi
6500     \else% Gin@ewidth empty
6501         \ifx\Gin@mhrepos\@empty
6502             \mhgraphics[#1]{#2}
6503         \else
6504             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6505         \fi
6506     \fi% Gin@ewidth empty
6507 }
6508 }{
6509     \ifx\Gin@ewidth\@empty
6510         \ifx\Gin@mhrepos\@empty
6511             \mhgraphics[width=\slidewidth,#1]{#2}
6512         \else
6513             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6514         \fi
6515         \ifx\Gin@mhrepos\@empty
6516             \mhgraphics[#1]{#2}
6517         \else
6518             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6519         \fi
6520     \fi% Gin@ewidth empty
6521 }
6522 \end{center}
6523 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6524 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6525 }
6526 } % 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.

```

6527 \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

```

6528 \AddToHook{begindocument}{
6529     \definecolor{green}{rgb}{0,.5,0}
6530     \definecolor{purple}{cmyk}{.3,1,0,.17}
6531 }

```

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`.

```

6532 % \def\STpresent#1{\textcolor{blue}{#1}}
6533 \def\defemph#1{\textcolor{magenta}{#1}}
6534 \def\symrefemph#1{\textcolor{cyan}{#1}}
6535 \def\compemph#1{\textcolor{blue}{#1}}
6536 \def\titleemph#1{\textcolor{blue}{#1}}
6537 \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.

```

6538 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6539 \def\smalltextwarning{
6540   \pgfuseimage{miko@small@dbend}
6541   \xspace
6542 }
6543 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6544 \newrobustcmd\textwarning{
6545   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6546   \xspace
6547 }
6548 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6549 \newrobustcmd\bigtextwarning{
6550   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6551   \xspace
6552 }

```

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

```

6553 \newrobustcmd\putgraphicsat[3]{
6554   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6555 }
6556 \newrobustcmd\putat[2]{
6557   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6558 }

```

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`.

```

6559 \bool_if:NT \c__notesslides_sectocframes_bool {
6560   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6561     \newcounter{chapter}\counterwithin*{section}{chapter}
6562   }{
6563     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6564       \newcounter{chapter}\counterwithin*{section}{chapter}
6565     }
6566   }
6567 }

```

`\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
6568 \def\part@prefix{}
6569 \@ifpackageloaded{document-structure}{}{
6570   \str_case:VnF \__notesslidesstopsect {
6571     {part}{
6572       \int_set:Nn \l_document_structure_section_level_int {0}
6573       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6574     \def\part@prefix{\arabic{chapter}.}
6575   }
6576   {chapter}{
6577     \int_set:Nn \l_document_structure_section_level_int {1}
6578     \def\thesection{\arabic{chapter}.\arabic{section}}
6579     \def\part@prefix{\arabic{chapter}.}
6580   }
6581 }{
6582   \int_set:Nn \l_document_structure_section_level_int {2}
6583   \def\part@prefix{}
6584 }
6585 }
6586
6587 \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 chooses the L^AT_EX sectioning macros according to `\section@level`.

sfragment

```

6588 \renewenvironment{sfragment}[2][]{
6589   \_document_structure_omgroup_args:n { #1 }
6590   \int_incr:N \l_document_structure_section_level_int
6591   \bool_if:NT \c__notesslides_sectocframes_bool {
6592     \stepcounter{slide}
6593     \begin{frame}[noframenumbering]
6594       \vfill\Large\centering
6595       \red{
6596         \ifcase\l_document_structure_section_level_int\or
6597           \stepcounter{part}
6598           \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6599           \def\currentsectionlevel{\omdoc@part@kw}
6600         \or
6601           \stepcounter{chapter}
6602           \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6603           \def\currentsectionlevel{\omdoc@chapter@kw}
6604         \or
6605           \stepcounter{section}
6606           \def\_notesslideslabel{\part@prefix\arabic{section}}
6607           \def\currentsectionlevel{\omdoc@section@kw}
6608         \or
6609           \stepcounter{subsection}
6610           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6611           \def\currentsectionlevel{\omdoc@subsection@kw}
6612         \or
6613           \stepcounter{subsubsection}
6614           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6615           \def\currentsectionlevel{\omdoc@subsubsection@kw}
6616         \or
6617           \stepcounter{paragraph}
6618           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6619           \def\currentsectionlevel{\omdoc@paragraph@kw}
6620       \else
6621         \def\_notesslideslabel{}

```

```

6622         \def\currentsectionlevel{\omdoc@paragraph@kw}
6623         \fi% end ifcase
6624         \_notesslideslabel%\sref@label@id\_notesslideslabel
6625         \quad #2%
6626     }%
6627     \vfill%
6628     \end{frame}%
6629 }
6630 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6631     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6632 }
6633 }{}
6634 }

```

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

```

6635 \def\inserttheorembodyfont{\normalfont}
6636 %\bool_if:NF \c__notesslides_notes_bool {
6637 % \defbeamertemplate{theorem begin}{miko}
6638 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6639 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6640 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6641 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6642 % \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.

```

6643 % \expandafter\def\csname Parent2\endcsname{}
6644 %}
6645
6646 \AddToHook{begindocument}{% this does not work for some reason
6647     \setbeamertemplate{theorems}[ams style]
6648 }
6649 \bool_if:NT \c__notesslides_notes_bool {
6650     \renewenvironment{columns}[1][{}]{%
6651         \par\noindent%
6652         \begin{minipage}%
6653             \slidewidth\centering\leavevmode%
6654     }{}%
6655     \end{minipage}\par\noindent%
6656 }%
6657 \newsavebox\columnbox%
6658 \renewenvironment<>{column}[2][{}]{%
6659     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6660 }{}%
6661     \end{minipage}\end{lrbox}\usebox\columnbox%
6662 }%
6663 }
6664 \bool_if:NFT \c__notesslides_noproblems_bool {
6665     \newenvironment{problems}{}{}
6666 }{
6667     \excludecomment{problems}
6668 }

```

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.

```

6669 \gdef\printexcursions{}
6670 \newcommand\excursionref[2]{% label, text
6671   \bool_if:NT \c__notesslides_notes_bool {
6672     \begin{sparagraph}[title=Excursion]
6673       #2 \sref[fallback=the appendix]{#1}.
6674     \end{sparagraph}
6675   }
6676 }
6677 \newcommand\activate@excursion[2][]{
6678   \gappto\printexcursions{\inputref{#1}{#2}}
6679 }
6680 \newcommand\excursion[4][]{% repos, label, path, text
6681   \bool_if:NT \c__notesslides_notes_bool {
6682     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6683   }
6684 }

```

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

`\excursiongroup`

```

6685 \keys_define:nn{notesslides / excursiongroup }{
6686   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6687   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6688   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6689 }
6690 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6691   \tl_clear:N \l__notesslides_excursion_intro_tl
6692   \str_clear:N \l__notesslides_excursion_id_str
6693   \str_clear:N \l__notesslides_excursion_mhrepos_str
6694   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6695 }
6696 \newcommand\excursiongroup[1][]{
6697   \__notesslides_excursion_args:n{ #1 }
6698   \ifdefempty\printexcursions{}% only if there are excursions
6699   {\begin{note}
6700     \begin{sfragment}[#1]{Excursions}%
6701     \ifdefempty\l__notesslides_excursion_intro_tl{
6702       \inputref[\l__notesslides_excursion_mhrepos_str]{
6703         \l__notesslides_excursion_intro_tl
6704       }
6705     }
6706     \printexcursions%
6707     \end{sfragment}
6708     \end{note}}
6709 }
6710 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6711 \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.

```
6712 <*package>
6713 <@@=problems>
6714 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6715 \RequirePackage{l3keys2e,stex}
6716
6717 \keys_define:nn { problem / pkg }{
6718   notes      .default:n   = { true },
6719   notes      .bool_set:N  = \c__problems_notes_bool,
6720   gnotes     .default:n   = { true },
6721   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
6722   hints      .default:n   = { true },
6723   hints      .bool_set:N  = \c__problems_hints_bool,
6724   solutions  .default:n   = { true },
6725   solutions  .bool_set:N  = \c__problems_solutions_bool,
6726   pts        .default:n   = { true },
6727   pts        .bool_set:N  = \c__problems_pts_bool,
6728   min        .default:n   = { true },
6729   min        .bool_set:N  = \c__problems_min_bool,
6730   boxed      .default:n   = { true },
6731   boxed      .bool_set:N  = \c__problems_boxed_bool,
6732   unknown    .code:n      = {}
6733 }
6734 \newif\ifsolutions
6735
6736 \ProcessKeysOptions{ problem / pkg }
6737 \bool_if:NTF \c__problems_solutions_bool {
6738   \solutionstrue
6739 }{
6740   \solutionsfalse
6741 }
```

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

```
6742 \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.

```
6743 \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.

```
6744 \def\prob@problem@kw{Problem}
6745 \def\prob@solution@kw{Solution}
6746 \def\prob@hint@kw{Hint}
6747 \def\prob@note@kw{Note}
6748 \def\prob@gnote@kw{Grading}
6749 \def\prob@pt@kw{pt}
6750 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6751 \AddToHook{begindocument}{
6752   \ltx@ifpackageloaded{babel}{
6753     \makeatletter
6754     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6755     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6756       \input{problem-ngerman.ldf}
6757     }
6758     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6759       \input{problem-finnish.ldf}
6760     }
6761     \clist_if_in:NnT \l_tmpa_clist {french}{
6762       \input{problem-french.ldf}
6763     }
6764     \clist_if_in:NnT \l_tmpa_clist {russian}{
6765       \input{problem-russian.ldf}
6766     }
6767     \makeatother
6768   }{}
6769 }
```

40.2 Problems and Solutions

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

```
6770 \keys_define:nn{ problem / problem }{
6771   id      .str_set:x:N = \l__problems_prob_id_str,
6772   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6773   min     .tl_set:N    = \l__problems_prob_min_tl,
6774   title   .tl_set:N    = \l__problems_prob_title_tl,
6775   type    .tl_set:N    = \l__problems_prob_type_tl,
6776   refnum  .int_set:N   = \l__problems_prob_refnum_int
6777 }
6778 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6779 \str_clear:N \l__problems_prob_id_str
6780 \tl_clear:N \l__problems_prob_pts_tl
6781 \tl_clear:N \l__problems_prob_min_tl
6782 \tl_clear:N \l__problems_prob_title_tl
6783 \tl_clear:N \l__problems_prob_type_tl
6784 \int_zero_new:N \l__problems_prob_refnum_int
6785 \keys_set:nn { problem / problem }{ #1 }
6786 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6787   \let\l__problems_prob_refnum_int\undefined
6788 }
6789 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6790 \newcounter{problem}
6791 \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.

```

6792 \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

```

6793 \newcommand\prob@number{
6794   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6795     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6796   }{
6797     \int_if_exist:NTF \l__problems_prob_refnum_int {
6798       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6799     }{
6800       \prob@label\theproblem
6801     }
6802   }
6803 }

```

(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.

```

6804 \newcommand\prob@title[3]{%
6805   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6806     #2 \l__problems_inclprob_title_tl #3
6807   }{
6808     \tl_if_exist:NTF \l__problems_prob_title_tl {
6809       #2 \l__problems_prob_title_tl #3
6810     }{
6811       #1
6812     }
6813   }
6814 }

```

(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.

```
6815 \def\prob@heading{
6816   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6817   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
6818 }
```

(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`

```
6819 \newenvironment{sproblem}[1][]{
6820   \__problems_prob_args:n{#1}%\sref@target%
6821   \@in@omtexttrue% we are in a statement (for inline definitions)
6822   \stepcounter{problem}\record@problem
6823   \def\current@section@level{\prob@problem@kw}
6824   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6825     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6826   }{
6827     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6828   }
6829   \str_if_exist:NTF \l__problems_inclprob_id_str {
6830     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6831   }{
6832     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6833   }
6834
6835
6836   \clist_set:No \l_tmpa_clist \sproblemtype
6837   \tl_clear:N \l_tmpa_tl
6838   \clist_map_inline:Nn \l_tmpa_clist {
6839     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6840       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6841     }
6842   }
6843   \tl_if_empty:NTF \l_tmpa_tl {
6844     \__problems_sproblem_start:
6845   }{
6846     \l_tmpa_tl
6847   }
6848   \stex_ref_new_doc_target:n \sproblemid
6849 }{
6850   \clist_set:No \l_tmpa_clist \sproblemtype
6851   \tl_clear:N \l_tmpa_tl
6852   \clist_map_inline:Nn \l_tmpa_clist {
6853     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6854       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6855     }
6856   }
```

```

6856 }
6857 \tl_if_empty:NTF \l_tmpa_tl {
6858   \__problems_sproblem_end:
6859 }{
6860   \l_tmpa_tl
6861 }
6862
6863
6864 \smallskip
6865 }
6866
6867
6868 \cs_new_protected:Nn \__problems_sproblem_start: {
6869   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6870 }
6871 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6872
6873 \newcommand\stexpatchproblem[3][] {
6874   \str_set:Nx \l_tmpa_str{ #1 }
6875   \str_if_empty:NTF \l_tmpa_str {
6876     \tl_set:Nn \__problems_sproblem_start: { #2 }
6877     \tl_set:Nn \__problems_sproblem_end: { #3 }
6878   }{
6879     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6880     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6881   }
6882 }
6883
6884
6885 \bool_if:NT \c__problems_boxed_bool {
6886   \surroundwithmdframed{problem}
6887 }

```

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

```

6888 \def\record@problem{
6889   \protected@write\@auxout{}
6890   {
6891     \string\@problem{\prob@number}
6892     {
6893       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6894         \l__problems_inclprob_pts_tl
6895       }{
6896         \l__problems_prob_pts_tl
6897       }
6898     }%
6899     {
6900       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6901         \l__problems_inclprob_min_tl
6902       }{
6903         \l__problems_prob_min_tl
6904       }
6905     }
6906   }
6907 }

```

(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).

```
6908 \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.

```
6909 \keys_define:nn { problem / solution }{
6910   id          .str_set_x:N = \l__problems_solution_id_str ,
6911   for         .tl_set:N    = \l__problems_solution_for_tl ,
6912   height      .dim_set:N   = \l__problems_solution_height_dim ,
6913   creators    .clist_set:N = \l__problems_solution_creators_clist ,
6914   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
6915   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
6916 }
6917 \cs_new_protected:Nn \__problems_solution_args:n {
6918   \str_clear:N \l__problems_solution_id_str
6919   \tl_clear:N \l__problems_solution_for_tl
6920   \tl_clear:N \l__problems_solution_srccite_tl
6921   \clist_clear:N \l__problems_solution_creators_clist
6922   \clist_clear:N \l__problems_solution_contributors_clist
6923   \dim_zero:N \l__problems_solution_height_dim
6924   \keys_set:nn { problem / solution }{ #1 }
6925 }
```

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

```
6926 \newcommand\@startsolution[1][{}]{
6927   \__problems_solution_args:n { #1 }
6928   \@in@omtexttrue% we are in a statement.
6929   \bool_if:NF \c__problems_boxed_bool { \hrule }
6930   \smallskip\noindent
6931   {\textbf\prob@solution@kw :\enspace}
6932   \begin{small}
6933   \def\current@section@level{\prob@solution@kw}
6934   \ignorespacesandpars
6935 }
```

\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.

```
6936 \newcommand\startsolutions{
6937   \specialcomment{solution}{\@startsolution}{
6938     \bool_if:NF \c__problems_boxed_bool {
6939       \hrule\medskip
6940     }
6941     \end{small}%
6942   }
6943   \bool_if:NT \c__problems_boxed_bool {
6944     \surroundwithmdframed{solution}
6945   }
6946 }
```

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

\stopsolutions

```
6947 \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.

```
6948 \ifsolutions
6949   \startsolutions
6950 \else
6951   \stopsolutions
6952 \fi
```

exnote

```
6953 \bool_if:NTF \c__problems_notes_bool {
6954   \newenvironment{exnote}[1][]{
6955     \par\smallskip\hrule\smallskip
6956     \noindent\textbf{\prob@note@kw : }\small
6957   }{
6958     \smallskip\hrule
6959   }
6960 }{
6961   \excludecomment{exnote}
6962 }
```

hint

```
6963 \bool_if:NTF \c__problems_notes_bool {
6964   \newenvironment{hint}[1][]{
6965     \par\smallskip\hrule\smallskip
6966     \noindent\textbf{\prob@hint@kw :~ }\small
6967   }{
6968     \smallskip\hrule
6969   }
6970 \newenvironment{exhint}[1][]{
6971   \par\smallskip\hrule\smallskip
6972   \noindent\textbf{\prob@hint@kw :~ }\small
6973 }{
6974   \smallskip\hrule
6975 }
6976 }{
6977   \excludecomment{hint}
6978   \excludecomment{exhint}
6979 }
```

gnote

```
6980 \bool_if:NTF \c__problems_notes_bool {
6981   \newenvironment{gnote}[1][]{
6982     \par\smallskip\hrule\smallskip
6983     \noindent\textbf{\prob@gnote@kw : }\small
6984   }{
6985     \smallskip\hrule
6986   }
6987 }{
6988   \excludecomment{gnote}
6989 }
```

40.3 Multiple Choice Blocks

EdN:22

mcb 22

```
6990 \newenvironment{mcb}{
6991   \begin{enumerate}
6992 }{
6993   \end{enumerate}
6994 }
```

we define the keys for the mcc macro

```
6995 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6996   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6997     \bool_set_true:N #1
6998   }{
6999     \bool_set_false:N #1
7000   }
7001 }
7002 \keys_define:nn { problem / mcc }{
7003   id          .str_set_x:N = \l__problems_mcc_id_str ,
7004   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7005   T           .default:n   = { true } ,
7006   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7007   F           .default:n   = { true } ,
7008   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7009   Ttext       .code:n      = {
7010     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7011   } ,
7012   Ftext       .code:n      = {
7013     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7014   }
7015 }
7016 \cs_new_protected:Nn \l__problems_mcc_args:n {
7017   \str_clear:N \l__problems_mcc_id_str
7018   \tl_clear:N \l__problems_mcc_feedback_tl
7019   \bool_set_true:N \l__problems_mcc_t_bool
7020   \bool_set_true:N \l__problems_mcc_f_bool
7021   \bool_set_true:N \l__problems_mcc_Ttext_bool
7022   \bool_set_false:N \l__problems_mcc_Ftext_bool
7023   \keys_set:nn { problem / mcc }{ #1 }
7024 }
```

\mcc

```
7025 \newcommand\mcc[2][] {
7026   \l__problems_mcc_args:n{ #1 }
7027   \item #2
7028   \ifsolutions
7029     \\\
7030     \bool_if:NT \l__problems_mcc_t_bool {
7031       % TODO!
7032       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
7033     }
7034     \bool_if:NT \l__problems_mcc_f_bool {
```

²²EdNOTE: MK: maybe import something better here from a dedicated MC package


```

7035         % TODO!
7036         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7037     }
7038     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7039         !
7040     }{
7041         \l__problems_mcc_feedback_tl
7042     }
7043     \fi
7044 } %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.

```

7045
7046 \keys_define:nn{ problem / inclproblem }{
7047   id      .str_set:N = \l__problems_inclprob_id_str,
7048   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7049   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7050   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7051   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7052   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7053   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7054 }
7055 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7056   \str_clear:N \l__problems_prob_id_str
7057   \tl_clear:N \l__problems_inclprob_pts_tl
7058   \tl_clear:N \l__problems_inclprob_min_tl
7059   \tl_clear:N \l__problems_inclprob_title_tl
7060   \tl_clear:N \l__problems_inclprob_type_tl
7061   \int_zero_new:N \l__problems_inclprob_refnum_int
7062   \str_clear:N \l__problems_inclprob_mhrepos_str
7063   \keys_set:nn { problem / inclproblem }{ #1 }
7064   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7065     \let\l__problems_inclprob_pts_tl\undefined
7066   }
7067   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7068     \let\l__problems_inclprob_min_tl\undefined
7069   }
7070   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7071     \let\l__problems_inclprob_title_tl\undefined
7072   }
7073   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7074     \let\l__problems_inclprob_type_tl\undefined
7075   }
7076   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7077     \let\l__problems_inclprob_refnum_int\undefined
7078   }
7079 }

```

```

7080
7081 \cs_new_protected:Nn \__problems_inclprob_clear: {
7082   \let\l__problems_inclprob_id_str\undefined
7083   \let\l__problems_inclprob_pts_tl\undefined
7084   \let\l__problems_inclprob_min_tl\undefined
7085   \let\l__problems_inclprob_title_tl\undefined
7086   \let\l__problems_inclprob_type_tl\undefined
7087   \let\l__problems_inclprob_refnum_int\undefined
7088   \let\l__problems_inclprob_mhrepos_str\undefined
7089 }
7090 \__problems_inclprob_clear:
7091
7092 \newcommand\includeproblem[2][ ]{
7093   \__problems_inclprob_args:n{ #1 }
7094   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7095     \input{#2}
7096   }{
7097     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7098       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7099     }
7100   }
7101   \__problems_inclprob_clear:
7102 }

```

(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.

```

7103 \AddToHook{enddocument}{
7104   \bool_if:NT \c__problems_pts_bool {
7105     \message{Total:~\arabic{pts}~points}
7106   }
7107   \bool_if:NT \c__problems_min_bool {
7108     \message{Total:~\arabic{min}~minutes}
7109   }
7110 }

```

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

```

7111 \def\pts#1{
7112   \bool_if:NT \c__problems_pts_bool {
7113     \marginpar{#1~\prob@pt@kw}
7114   }
7115 }
7116 \def\min#1{
7117   \bool_if:NT \c__problems_min_bool {
7118     \marginpar{#1~\prob@min@kw}
7119   }
7120 }

```

`\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.

```

7121 \newcounter{pts}
7122 \def\show@pts{
7123   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7124     \bool_if:NT \c__problems_pts_bool {
7125       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7126       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7127     }
7128   }{
7129     \tl_if_exist:NT \l__problems_prob_pts_tl {
7130       \bool_if:NT \c__problems_pts_bool {
7131         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7132         \addtocounter{pts}{\l__problems_prob_pts_tl}
7133       }
7134     }
7135   }
7136 }

```

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

`\show@min`

```

7137 \newcounter{min}
7138 \def\show@min{
7139   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7140     \bool_if:NT \c__problems_min_bool {
7141       \marginpar{\l__problems_inclprob_min_tl\ min}
7142       \addtocounter{min}{\l__problems_inclprob_min_tl}
7143     }
7144   }{
7145     \tl_if_exist:NT \l__problems_prob_min_tl {
7146       \bool_if:NT \c__problems_min_bool {
7147         \marginpar{\l__problems_prob_min_tl\ min}
7148         \addtocounter{min}{\l__problems_prob_min_tl}
7149       }
7150     }
7151   }
7152 }
7153 </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.

```
7154 \@@=hwexam>
7155 \*cls>
7156 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7157 \RequirePackage{l3keys2e}
7158 \DeclareOption*{
7159   \PassOptionsToClass{\CurrentOption}{document-structure}
7160   \PassOptionsToPackage{\CurrentOption}{stex}
7161   \PassOptionsToPackage{\CurrentOption}{hwexam}
7162   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7163 }
7164 \ProcessOptions
```

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

```
7165 \LoadClass{document-structure}
7166 \RequirePackage{stex}
7167 \RequirePackage{hwexam}
7168 \RequirePackage{tikzinput}
7169 \RequirePackage{graphicx}
7170 \RequirePackage{a4wide}
7171 \RequirePackage{amssymb}
7172 \RequirePackage{amstext}
7173 \RequirePackage{amsmath}
```

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

```

7174 \newcommand\assig@default@type{\hwexam@assignment@kw}
7175 \def\document@hwexamtype{\assig@default@type}
7176 <@@=document_structure>
7177 \keys_define:nn { document-structure / document }{
7178 id .str_set_x:N = \c_document_structure_document_id_str,
7179 hwexamtype .tl_set:N = \document@hwexamtype
7180 }
7181 <@@=hwexam>
7182 </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.

```
7183 \*package>
7184 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7185 \RequirePackage{13keys2e}
7186
7187 \newif\iftest\testfalse
7188 \DeclareOption{test}{\testtrue}
7189 \newif\ifmultiple\multiplefalse
7190 \DeclareOption{multiple}{\multipletrue}
7191 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7192 \ProcessOptions
```

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

```
7193 \RequirePackage{keyval}[1997/11/10]
7194 \RequirePackage{problem}
```

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

```
7195 \newcommand\hwexam@assignment@kw{Assignment}
7196 \newcommand\hwexam@given@kw{Given}
7197 \newcommand\hwexam@due@kw{Due}
7198 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7199 blank~for~extra~space}
7200 \def\hwexam@minutes@kw{minutes}
7201 \newcommand\correction@probs@kw{prob.}
7202 \newcommand\correction@pts@kw{total}
7203 \newcommand\correction@reached@kw{reached}
7204 \newcommand\correction@sum@kw{Sum}
7205 \newcommand\correction@grade@kw{grade}
7206 \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

```

7207 \AddToHook{begindocument}{
7208 \ltx@ifpackageloaded{babel}{
7209 \makeatletter
7210 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7211 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7212 \input{hwexam-ngerman.ldf}
7213 }
7214 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7215 \input{hwexam-finnish.ldf}
7216 }
7217 \clist_if_in:NnT \l_tmpa_clist {french}{
7218 \input{hwexam-french.ldf}
7219 }
7220 \clist_if_in:NnT \l_tmpa_clist {russian}{
7221 \input{hwexam-russian.ldf}
7222 }
7223 \makeatother
7224 }{}
7225 }
7226

```

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.

```

7227 \newcounter{assignment}
7228 \numberproblemsin{assignment}
7229 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7230 \keys_define:nn { hwexam / assignment } {
7231 id .str_set:N = \l__hwexam_assign_id_str,
7232 number .int_set:N = \l__hwexam_assign_number_int,
7233 title .tl_set:N = \l__hwexam_assign_title_tl,
7234 type .tl_set:N = \l__hwexam_assign_type_tl,
7235 given .tl_set:N = \l__hwexam_assign_given_tl,
7236 due .tl_set:N = \l__hwexam_assign_due_tl,
7237 loadmodules .code:n = {
7238 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7239 }
7240 }
7241 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7242 \str_clear:N \l__hwexam_assign_id_str
7243 \int_set:Nn \l__hwexam_assign_number_int {-1}
7244 \tl_clear:N \l__hwexam_assign_title_tl
7245 \tl_clear:N \l__hwexam_assign_type_tl
7246 \tl_clear:N \l__hwexam_assign_given_tl
7247 \tl_clear:N \l__hwexam_assign_due_tl
7248 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7249 \keys_set:nn { hwexam / assignment }{ #1 }
7250 }

```

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.

```

7251 \newcommand\given@due[2]{
7252 \bool_lazy_all:nF {
7253 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7254 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7255 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7256 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7257 }{ #1 }
7258
7259 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7260 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7261 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7262 }
7263 }{
7264 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7265 }
7266
7267 \bool_lazy_or:nnF {
7268 \bool_lazy_and_p:nn {
7269 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7270 }{
7271 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7272 }
7273 }{
7274 \bool_lazy_and_p:nn {
7275 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7276 }{
7277 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7278 }
7279 }{ ,~ }
7280
7281 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7282 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7283 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7284 }
7285 }{
7286 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7287 }
7288
7289 \bool_lazy_all:nF {
7290 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7291 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7292 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7293 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7294 }{ #2 }
7295 }

```

`\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.

```

7296 \newcommand\assignment@title[3]{
7297 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7298 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7299 #1
7300 }{
7301 #2\l__hwexam_assign_title_tl#3
7302 }
7303 }{
7304 #2\l__hwexam_inclasssign_title_tl#3
7305 }
7306 }

```

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

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

```

7307 \newcommand\assignment@number{
7308 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7309 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7310 \arabic{assignment}
7311 } {
7312 \int_use:N \l__hwexam_assign_number_int
7313 }
7314 }{
7315 \int_use:N \l__hwexam_inclasssign_number_int
7316 }
7317 }

```

(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.

```

7318 \newenvironment{assignment}[1][ ]{
7319 \__hwexam_assignment_args:n { #1 }
7320 %\sref@target
7321 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7322 \global\stepcounter{assignment}
7323 }{
7324 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7325 }
7326 \setcounter{problem}{0}
7327 \def\current@section@level{\document@hwexamtype}
7328 %\sref@label@id{\document@hwexamtype \thesection}
7329 \begin{@assignment}
7330 }{
7331 \end{@assignment}
7332 }

```

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

```

7333 \def\ass@title{
7334 \protect\document@hwexamtype~\arabic{assignment}
7335 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7336 }
7337 \ifmultiple
7338 \newenvironment{@assignment}{
7339 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7340 \begin{sfragment}[loadmodules]{\ass@title}
7341 }{
7342 \begin{sfragment}{\ass@title}
7343 }
7344 }{
7345 \end{sfragment}
7346 }

```

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

```

7347 \else
7348 \newenvironment{@assignment}{
7349 \begin{center}\bf
7350 \Large@title\strut\\
7351 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7352 \large\given@due{--;\}{}{;\}{}
7353 \end{center}
7354 }{}
7355 \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.

```

7356 \keys_define:nn { hwexam / inclassignment } {
7357 %id .str_set_x:N = \l__hwexam_assign_id_str,
7358 number .int_set:N = \l__hwexam_inclassign_number_int,
7359 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7360 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7361 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7362 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7363 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7364 }
7365 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7366 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7367 \tl_clear:N \l__hwexam_inclassign_title_tl
7368 \tl_clear:N \l__hwexam_inclassign_type_tl
7369 \tl_clear:N \l__hwexam_inclassign_given_tl
7370 \tl_clear:N \l__hwexam_inclassign_due_tl
7371 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7372 \keys_set:nn { hwexam / inclassignment }{ #1 }
7373 }
7374 \__hwexam_inclassignment_args:n {}
7375
7376 \newcommand\inputassignment[2][{}]{

```

```

7377 \_hwexam_inclassnment_args:n { #1 }
7378 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7379 \input{#2}
7380 }{
7381 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7382 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7383 }
7384 }
7385 \_hwexam_inclassnment_args:n {}
7386 }
7387 \newcommand\includeassignment[2][]{
7388 \newpage
7389 \inputassignment[#1]{#2}
7390 }

```

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

42.4 Typesetting Exams

\quizheading

```

7391 \ExplSyntaxOff
7392 \newcommand\quizheading[1]{%
7393 \def\@tas{#1}%
7394 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7395 \ifx\@tas\@empty\else%
7396 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7397 \fi%
7398 }
7399 \ExplSyntaxOn

```

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

\testheading

```

7400
7401 \def\hwexamheader{\input{hwexam-default.header}}
7402
7403 \def\hwexamminutes{
7404 \tl_if_empty:NTF \testheading@duration {
7405 {\testheading@min}~\hwexam@minutes@kw
7406 }{
7407 \testheading@duration
7408 }
7409 }
7410
7411 \keys_define:nn { hwexam / testheading } {
7412 min .tl_set:N = \testheading@min,
7413 duration .tl_set:N = \testheading@duration,
7414 reqpts .tl_set:N = \testheading@reqpts,
7415 tools .tl_set:N = \testheading@tools
7416 }
7417 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7418 \tl_clear:N \testheading@min
7419 \tl_clear:N \testheading@duration

```

```

7420 \tl_clear:N \testheading@reqpts
7421 \tl_clear:N \testheading@tools
7422 \keys_set:nn { hwexam / testheading }{ #1 }
7423 }
7424 \newenvironment{testheading}[1][]{
7425   \_hwexam_testheading_args:n{ #1 }
7426   \newcount\check@time\check@time=\testheading@min
7427   \advance\check@time by -\theassignment@totalmin
7428   \newif\if@bonuspoints
7429   \tl_if_empty:NTF \testheading@reqpts {
7430     \@bonuspointsfalse
7431   }{
7432     \newcount\bonus@pts
7433     \bonus@pts=\theassignment@totalpts
7434     \advance\bonus@pts by -\testheading@reqpts
7435     \edef\bonus@pts{\the\bonus@pts}
7436     \@bonuspointstrue
7437   }
7438   \edef\check@time{\the\check@time}
7439
7440   \makeatletter\hwexamheader\makeatother
7441 }{
7442   \newpage
7443 }

```

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

\testspace

```

7444 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7445 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7446 \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.

```

7447 <@=problems>
7448 \renewcommand\@problem[3]{
7449   \stepcounter{assignment@probs}
7450   \def\__problemspts{#2}
7451   \ifx\__problemspts\@empty\else
7452     \addtocounter{assignment@totalpts}{#2}
7453   \fi
7454   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7455   \xdef\correction@probs{\correction@probs & #1}%
7456   \xdef\correction@pts{\correction@pts & #2}
7457   \xdef\correction@reached{\correction@reached &}

```

```

7458 }
7459 <@@=hwexam>

```

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

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

```

7460 \newcounter{assignment@probs}
7461 \newcounter{assignment@totalpts}
7462 \newcounter{assignment@totalmin}
7463 \def\correction@probs{\correction@probs@kw}
7464 \def\correction@pts{\correction@pts@kw}
7465 \def\correction@reached{\correction@reached@kw}
7466 \stepcounter{assignment@probs}
7467 \newcommand\correction@table{
7468 \resizebox{\textwidth}{!}{%
7469 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7470 &\multicolumn{\theassignment@probs}{c|}|%|
7471 {\footnotesize\correction@forgrading@kw} &\\ \hline
7472 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7473 \correction@pts & \theassignment@totalpts & \\ \hline
7474 \correction@reached & & \[.7cm]\hline
7475 \end{tabular}}
7476 </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}

```