

The sTeX3 Package *

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

2022-02-19

Abstract

sTeX is a collection of L^AT_EX package that allow to markup documents semantically without leaving the document format, essentially turning L^AT_EX into a document format for mathematical knowledge management (MKM).

sTeX augments L^AT_EX with

- *Semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,
- A powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document,
- A mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

This is the full documentation of sTeX. It consists of four parts:

- **Part I** is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.
- **Part II** documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.
- **Part III** documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.
- **Part IV** is the detailed documentation of the sTeX package implementation.

*Version 3.0 (last revised 2022-02-19)

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)	10
6	sTeX Statements (Definitions, Theorems, Examples, ...)	11
7	Additional Packages	12
7.1	Modular Document Structuring	12
7.2	Slides and Course Notes	12
7.3	Homework, Problems and Exams	12
8	Stuff	13
8.1	Modules	13
8.1.1	Semantic Macros and Notations	13
	Other Argument Types	15
	Precedences	17
8.1.2	Archives and Imports	17
	Namespaces	17
	Paths in Import-Statements	18
II	Documentation	19
9	sTeX-Basics	20
9.1	Macros and Environments	20
9.1.1	HTML Annotations	20
9.1.2	Babel Languages	21
9.1.3	Auxiliary Methods	21

10	<code>sTeX</code>-MathHub	22
10.1	Macros and Environments	22
10.1.1	Files, Paths, URIs	22
10.1.2	MathHub Archives	23
10.1.3	Using Content in Archives	24
11	<code>sTeX</code>-References	25
11.1	Macros and Environments	25
11.1.1	Setting Reference Targets	25
11.1.2	Using References	26
12	<code>sTeX</code>-Modules	27
12.1	Macros and Environments	27
12.1.1	The <code>smodule</code> environment	29
13	<code>sTeX</code>-Module Inheritance	31
13.1	Macros and Environments	31
13.1.1	SMS Mode	31
13.1.2	Imports and Inheritance	32
14	<code>sTeX</code>-Symbols	35
14.1	Macros and Environments	35
15	<code>sTeX</code>-Terms	38
15.1	Macros and Environments	38
16	<code>sTeX</code>-Structural Features	41
16.1	Macros and Environments	41
16.1.1	Structures	41
17	<code>sTeX</code>-Statements	42
17.1	Macros and Environments	42
18	<code>sTeX</code>-Proofs: Structural Markup for Proofs	43
18.1	Introduction	45
18.2	The User Interface	46
18.2.1	Package Options	46
18.2.2	Proofs and Proof steps	46
18.2.3	Justifications	46
18.2.4	Proof Structure	47
18.2.5	Proof End Markers	48
18.2.6	Configuration of the Presentation	48
18.3	Limitations	48
19	<code>sTeX</code>-Metatheory	50
19.1	Symbols	50
III	Extensions	51

20	Tikzinput	52
20.1	Macros and Environments	52
21	document-structure: Semantic Markup for Open Mathematical Documents in \LaTeX	53
21.1	Introduction	53
21.2	The User Interface	54
21.2.1	Package and Class Options	54
21.2.2	Document Structure	54
21.2.3	Ignoring Inputs	56
21.2.4	Structure Sharing	56
21.2.5	Global Variables	56
21.2.6	Colors	57
21.3	Limitations	57
22	NotesSlides – Slides and Course Notes	58
22.1	Introduction	58
22.2	The User Interface	58
22.2.1	Package Options	58
22.2.2	Notes and Slides	59
22.2.3	Header and Footer Lines of the Slides	60
22.2.4	Frame Images	60
22.2.5	Colors and Highlighting	61
22.2.6	Front Matter, Titles, etc.	61
22.2.7	Excursions	61
22.2.8	Miscellaneous	62
22.3	Limitations	62
23	problem.sty: An Infrastructure for formatting Problems	63
23.1	Introduction	63
23.2	The User Interface	63
23.2.1	Package Options	63
23.2.2	Problems and Solutions	64
23.2.3	Multiple Choice Blocks	65
23.2.4	Including Problems	65
23.2.5	Reporting Metadata	65
23.3	Limitations	65
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	67
24.1	Introduction	68
24.2	The User Interface	68
24.2.1	Package and Class Options	68
24.2.2	Assignments	68
24.2.3	Typesetting Exams	68
24.2.4	Including Assignments	69
24.3	Limitations	69
IV	Implementation	71

25	STEX-Basics Implementation	72
25.1	The STeXDocument Class	72
25.2	Preliminaries	72
25.3	Messages and logging	73
25.4	HTML Annotations	74
25.5	Babel Languages	77
25.6	Auxiliary Methods	78
26	STEX-MathHub Implementation	79
26.1	Generic Path Handling	79
26.2	PWD and kpsewhich	81
26.3	File Hooks and Tracking	82
26.4	MathHub Repositories	83
26.5	Using Content in Archives	87
27	STEX-References Implementation	92
27.1	Document URIs and URLs	92
27.2	Setting Reference Targets	94
27.3	Using References	96
28	STEX-Modules Implementation	99
28.1	The smodule environment	103
28.2	Invoking modules	108
29	STEX-Module Inheritance Implementation	110
29.1	SMS Mode	110
29.2	Inheritance	113
30	STEX-Symbols Implementation	117
30.1	Symbol Declarations	117
30.2	Notations	124
30.3	Variables	134
31	STEX-Terms Implementation	137
31.1	Symbol Invocations	137
31.2	Terms	140
31.3	Notation Components	147
32	STEX-Structural Features Implementation	150
32.1	Imports with modification	150
32.2	The feature environment	157
32.3	Features	158
33	STEX-Statements Implementation	164
33.1	Definitions	164
33.2	Assertions	169
33.3	Examples	172
33.4	Logical Paragraphs	174

34 The Implementation	180
34.1 Package Options	180
34.2 Proofs	180
34.3 Justifications	186
35 \LaTeX-Others Implementation	188
36 \LaTeX-Metatheory Implementation	189
37 Tikzinput Implementation	192
38 document-structure.sty Implementation	194
38.1 The document-structure Class	194
38.2 Class Options	194
38.3 Beefing up the <code>document</code> environment	195
38.4 Implementation: document-structure Package	195
38.5 Package Options	195
38.6 Document Structure	197
38.7 Front and Backmatter	200
38.8 Global Variables	202
39 NotesSlides – Implementation	203
39.1 Class and Package Options	203
39.2 Notes and Slides	205
39.3 Header and Footer Lines	209
39.4 Frame Images	210
39.5 Colors and Highlighting	211
39.6 Sectioning	212
39.7 Excursions	214
40 The Implementation	216
40.1 Package Options	216
40.2 Problems and Solutions	217
40.3 Multiple Choice Blocks	223
40.4 Including Problems	224
40.5 Reporting Metadata	225
41 Implementation: The hwexam Class	227
41.1 Class Options	227
42 Implementation: The hwexam Package	229
42.1 Package Options	229
42.2 Assignments	230
42.3 Including Assignments	233
42.4 Typesetting Exams	234
42.5 Leftovers	236

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Compiling this document with `pdflatex` should yield the output

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

Note that the \sum and ∞ -symbols are highlighted in blue, and the words “series” and “converges” in bold. This signifies that these words and symbols reference $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ *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. `\realdivide` and `\realpower`.

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using sTeX

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

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

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

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

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

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

Module 2:
Module 3:
Module 4:

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

Example 3

```
\begin{smodule}{ring}
\begin{copymodule}{group}{addition}
\renamedcl[name=universe]{universe}{runiverse}
\renamedcl[name=plus]{operation}{rplus}
\renamedcl[name=zero]{unit}{rzero}
\renamedcl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation*[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation*[zero]{rzero}{\comp0}
\notation*[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation*[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation*[one]{rone}{\comp1}
Test: $\rtimes a{\rplus c}{\rtimes de}$
\end{smodule}
```

Module 5: Test: $a \cdot (c + d \cdot e)$

TODO: explain donotclone

Example 4

```
\begin{smodule}{int}
\symdef{Integers}{\comp{\mathbb Z}}
\symdef[args=2,op=+]{plus}{#1 \comp+ #2}
\symdef{zero}{\comp0}
\symdef[args=1,op=-]{uminus}{\comp-#1}

\begin{interpretmodule}{group}{intisgroup}
\assign{universe}{\Integers}
\assign{operation}{\plus!}
\assign{unit}{\zero}
\assign{inverse}{\uminus!}
\end{interpretmodule}
\end{smodule}
```

Module 6:

5.2 Primitive Symbols (The sTeX Metatheory)

Chapter 6

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

Chapter 7

Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

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

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 5

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

ab

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

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

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

Example 6

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

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

.

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

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

Example 7

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

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

.

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

Example 8

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

Multiplying again by b yields...

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

Example 9

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

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

⁴EdNOTE: TODO

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

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

Example 10

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

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

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

Example 11

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

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

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

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

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

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

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

Other Argument Types

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

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

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

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

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

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

Example 12

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

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

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

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

Example 13

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

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

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

5 6

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

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

Precedences

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

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

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

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

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

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

For example:

Module 9:

Example 14

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

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

8.1.2 Archives and Imports

Namespaces

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

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

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 9

sTeX-Basics

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

`\sref` `\sref[<opt-args>]{<id>}`

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

`\srefsym` `\srefsym[<opt-args>]{<symbol>}`

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>`.

`\srefsymuri` `\srefsymuri{<URI>}{<text>}`

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:

 $\backslash\text{c_stex_module_}\langle URI \rangle_prop$

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.

 $\backslash\text{c_stex_module_}\langle URI \rangle_code$

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

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

The names of all constants declared in the module

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

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:

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 1

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 2

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

```
Module 10:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
                Meaning: >macro:->\protect \bar  <
Module 11:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 12:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

`\usemodule`

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

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

Test 3

```

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

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

```

Module 13:

Module 14: Meaning: »macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}«

Module 15: Meaning: »undefined«

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

All modules: [http://mathhub.info/sTeX?Metatheory, file://stextest?UseTest3, file://stextest?UseTest2](http://mathhub.info/sTeX?Metatheory,file://stextest?UseTest3,file://stextest?UseTest2)

All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?aseqdots>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, [http://mathhub.info/sTeX?Metatheory?module-type](http://mathhub.info/sTeX?Metatheory?let, <a href=), <http://mathhub.info/sTeX?Metatheory?mat>, <http://mathhub.info/sTeX?Metatheory?structure>, <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dur>, <http://mathhub.info/sTeX?Metatheory?proposition>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?aseqdots>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?let>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mat>, <http://mathhub.info/sTeX?Metatheory?structure>, file://stextest?UseTest2?bar

Test 4

```

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

```

Circular dependencies:

Module 16: »macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}«

»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}«

`\stex_import_module_uri:nn`

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

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

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

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

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

That module should have the same namespace as the current one.

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

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

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

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

Chapter 14

TeX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

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

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. $\backslash\text{symdecl}[\text{args=ii}]\{\text{plus}\}$ 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. $\backslash\text{symdecl}[\text{args=a}]\{\text{plus}\}$ 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. $\backslash\text{symdecl}[\text{args=bi}]\{\text{forall}\}$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

`\stex_symdecl_do:n`

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

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

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

Test 5

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

```
Module 17:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 6

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

Module 18:

`\symdef`

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

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

Test 7

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

Module 19: $a+b+c$

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {<body>}`

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

`\withbrackets`

`\withbrackets <left> <right> {<body>}`

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

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

Test 8

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

Module 20: $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 9

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

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{##1 \comp+ ##2}
\notation[prec=100]{ mult }{#1}{##1 \comp\cdot ##2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}]\$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[]{$\displaystyle
\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$}
\end{smodule}
```

Module 21: $\langle a|[b;c;d;e;f]^g \rangle$ and $\langle a|[b;c]^g \rangle$ and $\langle a|[b]^c \rangle$

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

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

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

`\stex_term_custom:nn`

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

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

Test 10

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

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

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

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

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

\bar![bar]

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

\end{smodule}
```

Module 22: some a and some b and also some c here.
 some a and some b and also some c here.
 bar
 or just some c
 bar
 or first b, then c, and finally a

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

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

`\compemph@uri`

`\defemph`

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

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 16

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

TeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

18.2.4 Proof Structure

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

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

21.1 Introduction

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

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

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

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

21.2 The User Interface

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

21.2.1 Package and Class Options

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

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

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

21.2.2 Document Structure

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

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

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

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

`\STRsemantics`

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

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

22.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

22.2.3 Header and Footer Lines of the Slides

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

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

22.2.4 Frame Images

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

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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

22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

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

Name:

320101 General Computer Science (Fall 2010)

2022-02-19

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.

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.

You have ample time, so take it slow and avoid rushing to mistakes!

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

[illegible]

good luck

Example 8: A generated test heading.

Part IV
Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

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

```

(End definition for `\stex` and `\TeX`. These functions are documented on page 20.)

25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

(End definition for `\stex_debug:nn`. This function is documented on page 20.)

Redirecting messages:

```

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

```

25.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

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

```

(End definition for `\if@latexml`. This function is documented on page 20.)

`\latexml_if_p:`

`\latexml_if:TF`

```

83 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
84   \if@latexml
85     \prg_return_true:
86   \else:
87     \prg_return_false:
88   \fi:
89 }

```

(End definition for `\latexml_if:TF`. This function is documented on page 20.)

`\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.

```

90 \tl_new:N \l__stex_annotate_arg_tl
91 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
92   \rustex_if:TF {
93     \rustex_direct_HTML:n { \c_ampsand_str lrm; }
94   }{-}
95 }

```

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

`_stex_annotate_checkempty:n`

```

96 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
97   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
98   \tl_if_empty:NT \l__stex_annotate_arg_tl {
99     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
100   }
101 }

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

102 \bool_new:N \_stex_html_do_output_bool
103 \bool_set_true:N \_stex_html_do_output_bool
104
105 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
106   \bool_if:nTF \_stex_html_do_output_bool
107     \prg_return_true: \prg_return_false:
108 }

```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 20.)

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

109 \cs_new_protected:Nn \stex_suppress_html:n {
110   \exp_args:Nne \use:nn {
111     \bool_set_false:N \_stex_html_do_output_bool
112     #1
113   }{
114     \stex_if_do_html:T {
115       \bool_set_true:N \_stex_html_do_output_bool
116     }
117   }
118 }

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 20.)

`\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`

```

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

```

```

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

```

```

188         \tl_use:N \l__stex_annotate_arg_tl
189     }
190 } {
191     \cs:w latexml@invisible@text\cs_end:{
192         \tl_use:N \l__stex_annotate_arg_tl
193     }
194 }
195 }
196 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
197     \__stex_annotate_checkempty:n { #3 }
198     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
199         \tl_use:N \l__stex_annotate_arg_tl
200     }
201 }
202 \NewDocumentEnvironment{stex_annotate_env} { m m } {
203     \par\begin{latexml@annotateenv}{#1}{#2}
204 }{
205     \par\end{latexml@annotateenv}
206 }
207 }{
208     \cs_new_protected:Nn \stex_annotate:nnn {#3}
209     \cs_new_protected:Nn \stex_annotate_invisible:n {}
210     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
211     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
212 }
213 }

```

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

25.5 Babel Languages

```

214 <@=stex_language>

```

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

```

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

```

```

234   romanian = ro ,
235   turkish  = tr ,
236   french   = fr
237 }
238 % todo: chinese simplified (zhs)
239 %       chinese traditional (zht)

```

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

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

```

240 \clist_if_empty:NF \c_stex_languages_clist {
241   \clist_clear:N \l_tmpa_clist
242   \clist_map_inline:Nn \c_stex_languages_clist {
243     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
244       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
245     } {
246       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
247     }
248   }
249   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
250   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
251 }

```

25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

252 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
253   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
254   \def#1{
255     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
256   }
257 }

```

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

`\stex_reactivate_macro:N`

```

258 \cs_new_protected:Nn \stex_reactivate_macro:N {
259   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
260 }

```

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

`\ignorespacesandpars`

```

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

```

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

Chapter 26

STEX -MathHub Implementation

```
270 <*package>
271
272 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
273
274 <@@=stex_path>
275
276 Warnings and error messages
277 \msg_new:nnn{stex}{error/norepository}{
278   No~archive~#1~found~in~#2
279 }
280 \msg_new:nnn{stex}{error/notinarchive}{
281   Not~currently~in~an~archive,~but~\detokenize{#1}~
282   needs~one!
283 }
284 \msg_new:nnn{stex}{error/nofile}{
285   \detokenize{#1}~could~not~find~file~#2
286 }
287 \msg_new:nnn{stex}{error/twofiles}{
288   \detokenize{#1}~found~two~candidates~for~#2
289 }
290 }
```

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`

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

```

296     \seq_map_inline:Nn #1 {
297       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
298       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
299     }
300     \seq_set_eq:NN #1 \l_tmpa_tl
301   }
302   \stex_path_canonicalize:N #1
303 }
304 }
305

```

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

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

```

306 \cs_new_protected:Nn \stex_path_to_string:NN {
307   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
308 }
309
310 \cs_new:Nn \stex_path_to_string:N {
311   \seq_use:Nn #1 /
312 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 22.)

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

```

313 \str_const:Nn \c__stex_path_dot_str {.}
314 \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.

```

315 \cs_new_protected:Nn \stex_path_canonicalize:N {
316   \seq_if_empty:NF #1 {
317     \seq_clear:N \l_tmpa_seq
318     \seq_get_left:NN #1 \l_tmpa_tl
319     \str_if_empty:NT \l_tmpa_tl {
320       \seq_put_right:Nn \l_tmpa_seq {}
321     }
322     \seq_map_inline:Nn #1 {
323       \str_set:Nn \l_tmpa_tl { ##1 }
324       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
325         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
326           \seq_if_empty:NNTF \l_tmpa_seq {
327             \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
328               \c__stex_path_up_str
329             }
330           }{
331             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
332             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
333               \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
334                 \c__stex_path_up_str
335               }
336             }{

```

```

337         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
338     }
339 }
340 }{
341     \str_if_empty:NF \l_tmpa_tl {
342         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
343     }
344 }
345 }
346 }
347 \seq_gset_eq:NN #1 \l_tmpa_seq
348 }
349 }

```

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

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

```

350 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
351     \seq_if_empty:NTF #1 {
352         \prg_return_false:
353     }{
354         \seq_get_left:NN #1 \l_tmpa_tl
355         \str_if_empty:NTF \l_tmpa_tl {
356             \prg_return_true:
357         }{
358             \prg_return_false:
359         }
360     }
361 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 22.)

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

362 \str_new:N\l_stex_kpsewhich_return_str
363 \cs_new_protected:Nn \stex_kpsewhich:n {
364     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
365     \exp_args:NNo \str_set:Nn \l_stex_kpsewhich_return_str{\l_tmpa_tl}
366     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
367 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

368 \sys_if_platform_windows:TF{
369     \begingroup\escapechar=-1\catcode'\=12
370     \exp_args:Nx \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
371     \exp_args:NNx \str_replace_all:Nnn \l_stex_kpsewhich_return_str{\c_backslash_str}/
372     \exp_args:Nnx \use:nn{\endgroup}{\str_set:Nn \exp_not:N \l_stex_kpsewhich_return_str{\l_stex_
373 }{
374     \stex_kpsewhich:n{-var-value~PWD}

```

```

375 }
376
377 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
378 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
379 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

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

26.3 File Hooks and Tracking

```

380 <@@=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 `STEX`-purposes.

```

\g_stex_files_stack keeps track of file changes
381 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

```

\c_stex_mainfile_seq
\c_stex_mainfile_str
382 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
383 \stex_path_from_string:Nn \c_stex_mainfile_seq
384 \c_stex_mainfile_str

```

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

```

\g_stex_currentfile_seq
385 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

```

\stex_filestack_push:n
386 \cs_new_protected:Nn \stex_filestack_push:n {
387   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
388   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
389     \stex_path_from_string:Nn\g_stex_currentfile_seq{
390       \c_stex_pwd_str/#1
391     }
392   }
393   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
394   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
395 }

```

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

`\stex_filestack_pop:`

```

396 \cs_new_protected:Nn \stex_filestack_pop: {
397   \seq_if_empty:NF\g__stex_files_stack{
398     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
399   }
400   \seq_if_empty:NTF\g__stex_files_stack{
401     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
402   }{
403     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
404     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
405   }
406 }

```

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

Hooks for the current file:

```

407 \AddToHook{file/before}{
408   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
409 }
410 \AddToHook{file/after}{
411   \stex_filestack_pop:
412 }

```

26.4 MathHub Repositories

```

413 <@=stex_mathhub>

```

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

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

```

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

```

```

439 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
440 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 23.)

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

```

441 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
442   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
443     \str_set:Nx \l_tmpa_str { #1 }
444     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
445     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
446     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
447     \__stex_mathhub_find_manifest:N \l_tmpa_seq
448     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
449       \msg_error:nnxx{stex}{error/norepository}{#1}{
450         \stex_path_to_string:N \c_stex_mathhub_str
451       }
452     } {
453       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
454     }
455   }
456 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

457 \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`:

```

458 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
459   \seq_set_eq:NN\l_tmpa_seq #1
460   \bool_set_true:N\l_tmpa_bool
461   \bool_while_do:Nn \l_tmpa_bool {
462     \seq_if_empty:NTF \l_tmpa_seq {
463       \bool_set_false:N\l_tmpa_bool
464     } {
465       \file_if_exist:nTF{
466         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
467       } {
468         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
469         \bool_set_false:N\l_tmpa_bool
470       } {
471         \file_if_exist:nTF{
472           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
473         } {
474           \seq_put_right:Nn\l_tmpa_seq{META-INF}
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       \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
486     }
487   }
488 }
489 }
490 }
491 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
492 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

493 \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:

```

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

```

```

524         { ns } \l_tmpb_tl
525     }
526     {dependencies} {
527         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528         { deps } \l_tmpb_tl
529     }
530     }{}{}
531 }{}
532 }
533 \ior_close:N \c__stex_mathhub_manifest_ior
534 }

```

(End definition for __stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```

535 \cs_new_protected:Nn \stex_set_current_repository:n {
536     \stex_require_repository:n { #1 }
537     \prop_set_eq:Nc \l_stex_current_repository_prop {
538         c_stex_mathhub_#1_manifest_prop
539     }
540 }

```

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

\stex_require_repository:n

```

541 \cs_new_protected:Nn \stex_require_repository:n {
542     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
543         \stex_debug:nn{mathhub}{Opening~archive:~#1}
544         \__stex_mathhub_do_manifest:n { #1 }
545     }
546 }

```

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

\l_stex_current_repository_prop Current MathHub repository

```

547 %\prop_new:N \l_stex_current_repository_prop
548
549 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
550 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
551     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
552 } {
553     \__stex_mathhub_parse_manifest:n { main }
554     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
555     \l_tmpa_str
556     \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
557     \c_stex_mathhub_main_manifest_prop
558     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
559     \stex_debug:nn{mathhub}{Current~repository:~
560         \prop_item:Nn \l_stex_current_repository_prop {id}
561     }
562 }

```

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

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

```

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

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [23](#).)

26.5 Using Content in Archives

`\mhpath`

```

601 \def \mhpath #1 #2 {
602   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
603     \c_stex_mathhub_str /
604     \prop_item:Nn \l_stex_current_repository_prop { id }
605     / source / #2
606   }{
607     \c_stex_mathhub_str / #1 / source / #2

```

```

608 }
609 }

```

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

`\inputref`
`\mhinput`

```

610 \newif \ifinputref \inputreffalse
611
612 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
613   \stex_in_repository:nn {#1} {
614     \ifinputref
615       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
616     \else
617       \inputreftrue
618       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
619     \inputreffalse
620   \fi
621 }
622 }
623 \NewDocumentCommand \mhinput { 0{} m}{
624   \stex_mhinput:nn{ #1 }{ #2 }
625 }
626
627 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
628   \stex_in_repository:nn {#1} {
629     \bool_lazy_any:nTF {
630       {\rustex_if_p:}
631       {\latexml_if_p:}
632     } {
633       \str_clear:N \l_tmpa_str
634       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
635         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
636       }
637       \stex_annotate_invisible:nnn{inputref}{
638         \l_tmpa_str / #2
639       }{}
640     }{
641       \begingroup
642         \inputreftrue
643         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
644       \endgroup
645     }
646   }
647 }
648 \NewDocumentCommand \inputref { 0{} m}{
649   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
650 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 24.)

`\addmhbibresource`

```

651 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
652   \stex_in_repository:nn {#1} {
653     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
654   }

```

```

655 }
656 \newcommand\addmhbibresource[2][]{
657   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
658 }

```

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

\libinput

```

659 \cs_new_protected:Npn \libinput #1 {
660   \prop_if_exist:NF \l_stex_current_repository_prop {
661     \msg_error:nnn{stex}{error/notinarchive}\libinput
662   }
663   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
664     \msg_error:nnn{stex}{error/notinarchive}\libinput
665   }
666   \seq_clear:N \l__stex_mathhub_libinput_files_seq
667   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
668   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
669
670   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
671     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
672     \IfFileExists{ \l_tmpa_str }{
673       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
674     }{}
675     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
676     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
677   }
678
679   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
680   \IfFileExists{ \l_tmpa_str }{
681     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
682   }{}
683
684   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
685     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
686   }{
687     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
688       \input{ ##1 }
689     }
690   }
691 }

```

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

\libusepackage

```

692 \NewDocumentCommand \libusepackage {0{} m} {
693   \prop_if_exist:NF \l_stex_current_repository_prop {
694     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
695   }
696   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
697     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
698   }
699   \tl_clear:N \l__stex_mathhub_libinput_files_seq
700   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
701   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

702
703 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
704   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
705   \IfFileExists{ \l_tmpa_str }{
706     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
707   }{}
708   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
709   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
710 }
711
712 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
713 \IfFileExists{ \l_tmpa_str }{
714   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
715 }{}
716
717 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
718   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
719 }{
720   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
721     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
722       \usepackage[#1]{ #1 }
723     }
724   }{
725     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
726   }
727 }
728 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

729
730 \AddToHook{begindocument}{
731   \ltx@ifpackageloaded{graphicx}{
732     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
733     \newcommand\mhgraphics[2][]{%
734       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
735       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
736     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
737   }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 24.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```


(End definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 24.)

Chapter 27

STEX -References Implementation

```
748 <*package>
749
750 %%%%%%%%%% references.dtx %%%%%%%%%%
751
752 <@@=stex_refs>
753
754 Warnings and error messages
```

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

```
754 \iow_new:N \c__stex_refs_refs_iow
755 \AddToHook{begindocument}{
756   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
757 }
758 \AddToHook{enddocument}{
759   \iow_close:N \c__stex_refs_refs_iow
760 }
```

`\STEXreftitle`

```
761 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
762
763 \NewDocumentCommand \STEXreftitle { m } {
764   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
765 }
```

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

27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
766 \str_new:N \l_stex_current_docns_str
```

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

`\stex_get_document_uri:`

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

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

`\l_stex_current_docurl_str`

```
807 \str_new:N \l_stex_current_docurl_str
```

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

`\stex_get_document_url:`

```
808 \cs_new_protected:Nn \stex_get_document_url: {  
809   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
810   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
811   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

812 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
813 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
814
815 \str_clear:N \l_tmpa_str
816 \prop_if_exist:NT \l_stex_current_repository_prop {
817   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
818     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
819       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
820     }
821   }
822 }
823
824 \str_if_empty:NTF \l_tmpa_str {
825   \str_set:Nx \l_stex_current_docurl_str {
826     file:/\stex_path_to_string:N \l_tmpa_seq
827   }
828 }{
829   \bool_set_true:N \l_tmpa_bool
830   \bool_while_do:Nn \l_tmpa_bool {
831     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
832     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
833       {source} { \bool_set_false:N \l_tmpa_bool }
834     }{}{
835       \seq_if_empty:NT \l_tmpa_seq {
836         \bool_set_false:N \l_tmpa_bool
837       }
838     }
839   }
840
841   \seq_if_empty:NTF \l_tmpa_seq {
842     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
843   }{
844     \str_set:Nx \l_stex_current_docurl_str {
845       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
846     }
847   }
848 }
849 }

```

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

27.2 Setting Reference Targets

```

850 \str_const:Nn \c__stex_refs_url_str{URL}
851 \str_const:Nn \c__stex_refs_ref_str{REF}
852 \str_new:N \l__stex_refs_curr_label_str
853 % @currentlabel -> number
854 % @currentlabelname -> title
855 % @currentHref -> name.number <- id of some kind
856 % \theH# -> \arabic{section}
857 % \the# -> number
858 % \hyper@makecurrent{#}
859 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

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

```

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

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

```

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

```

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

```

898 \AtEndDocument{
899   \def\stexauxadddocref#1 #2 {}{}
900 }

```

`\stex_ref_new_sym_target:n`

```

901 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
902   \stex_if_smsmode:TF {
903     \str_if_exist:cF{sref_sym_#1_type}{
904       \stex_get_document_url:
905       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

906     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
907   }
908 }{
909   \str_if_empty:NF \l__stex_refs_curr_label_str {
910     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
911     \immediate\write\@auxout{
912       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
913         \l__stex_refs_curr_label_str
914     }
915   }
916 }
917 }
918 }

```

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

27.3 Using References

```

919 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

920
921 \keys_define:nn { stex / sref } {
922   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
923   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
924   pre           .tl_set:N = \l__stex_refs_pre_tl ,
925   post          .tl_set:N = \l__stex_refs_post_tl ,
926 }
927 \cs_new_protected:Nn \__stex_refs_args:n {
928   \tl_clear:N \l__stex_refs_linktext_tl
929   \tl_clear:N \l__stex_refs_fallback_tl
930   \tl_clear:N \l__stex_refs_pre_tl
931   \tl_clear:N \l__stex_refs_post_tl
932   \str_clear:N \l__stex_refs_repo_str
933   \keys_set:nn { stex / sref } { #1 }
934 }

```

The actual macro:

```

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

```

```

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

```

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

\srefsym

```

997 \NewDocumentCommand \srefsym { 0{} m}{
998   \stex_get_symbol:n { #2 }
999   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1000 }

```

```

1001
1002 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1003   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1004     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1005   }{
1006     \__stex_refs_args:n { #1 }
1007     \str_if_empty:NTF \l__stex_refs_indocument_str {
1008       \tl_if_exist:cTF{sref_sym_#2 _type}{
1009         % doc uri in \l_tmpb_str
1010         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1011         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1012           % reference
1013           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1014             \cs_if_exist:cTF{autoref}{
1015               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1016             }{
1017               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1018             }
1019           }{
1020             \ltx@ifpackageloaded{hyperref}{
1021               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1022             }{
1023               \l__stex_refs_linktext_tl
1024             }
1025           }
1026         }{
1027           % URL
1028           \ltx@ifpackageloaded{hyperref}{
1029             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1030           }{
1031             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1032           }
1033         }
1034       }{
1035         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1036       }
1037     }{
1038       % TODO
1039     }
1040   }
1041 }

```

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

\srefsymuri

```

1042 \cs_new_protected:Npn \srefsymuri #1 #2 {
1043   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1044 }

```

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

```

1045 </package>

```


Chapter 28

STEX -Modules Implementation

```
1046 <*package>
1047
1048 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1049
1050 <@@=stex_modules>
1051
1052   Warnings and error messages
1053   \msg_new:nnn{stex}{error/unknownmodule}{
1054     No~module~#1~found
1055   }
1056   \msg_new:nnn{stex}{error/syntax}{
1057     Syntax~error:~#1
1058   }
1059   \msg_new:nnn{stex}{error/siglanguage}{
1060     Module~#1~declares~signature~#2,~but~does~not~
1061     declare~its~language
1062   }
1063   \msg_new:nnn{stex}{warning/deprecated}{
1064     #1~is~deprecated;~please~use~#2~instead!
1065   }
1066   \msg_new:nnn{stex}{error/conflictingmodules}{
1067     Conflicting~imports~for~module~#1
1068   }
1069
1070 \l_stex_current_module_str The current module:
1071
1072 \str_new:N \l_stex_current_module_str
1073
1074 (End definition for \l_stex_current_module_str. This variable is documented on page 28.)
1075
1076 \l_stex_all_modules_seq Stores all available modules
1077
1078 \seq_new:N \l_stex_all_modules_seq
1079
1080 (End definition for \l_stex_all_modules_seq. This variable is documented on page 28.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1070 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1071   \str_if_empty:NTF \l_stex_current_module_str
1072   \prg_return_false: \prg_return_true:
1073 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1074 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1075   \prop_if_exist:cTF { c_stex_module_#1_prop }
1076   \prg_return_true: \prg_return_false:
1077 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 28.)

\stex_add_to_current_module:n Only allowed within modules:

```

\STEXexport
1078 \cs_new_protected:Nn \stex_add_to_current_module:n {
1079   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1080 }
1081 \cs_new_protected:Npn \STEXexport {
1082   \begingroup
1083   \newlinechar=-1\relax
1084   \endlinechar=-1\relax
1085   %\catcode'\ = 9\relax
1086   \expandafter\endgroup\__stex_modules_export:n
1087 }
1088 \cs_new_protected:Nn \__stex_modules_export:n {
1089   \ignorespaces #1
1090   \stex_add_to_current_module:n { \ignorespaces #1 }
1091   \stex_smsmode_do:
1092 }
1093 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 28.)

```

\stex_add_constant_to_current_module:n
1094 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1095   \str_set:Nx \l_tmpa_str { #1 }
1096   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1097 }

```

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page 28.)

```

\stex_add_import_to_current_module:n
1098 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1099   \str_set:Nx \l_tmpa_str { #1 }
1100   \exp_args:Nno
1101   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1102     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1103   }
1104 }

```

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

`\stex_collect_imports:n`

```

1105 \cs_new_protected:Nn \stex_collect_imports:n {
1106   \seq_clear:N \l_stex_collect_imports_seq
1107   \__stex_modules_collect_imports:n {#1}
1108 }
1109 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1110   \seq_map_inline:cn {c_stex_module_#1_imports} {
1111     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1112       \__stex_modules_collect_imports:n { ##1 }
1113     }
1114   }
1115   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1116     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1117   }
1118 }

```

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

`\stex_do_up_to_module:n`

```

1119 \int_new:N \l__stex_modules_group_depth_int
1120 \tl_new:N \l__stex_modules_aftergroup_tl
1121 \cs_new_protected:Nn \stex_do_up_to_module:n {
1122   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1123     #1
1124   }{
1125     #1
1126     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1127       \aftergroup\__stex_modules_aftergroup_do:
1128     }
1129   }
1130   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1131     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1132       \l__stex_modules_aftergroup_tl
1133       \tl_clear:N \l__stex_modules_aftergroup_tl
1134     }{
1135       \l__stex_modules_aftergroup_tl
1136       \aftergroup\__stex_modules_aftergroup_do:
1137     }
1138   }

```

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

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

1139

(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.

```

1140 \str_new:N \l_stex_modules_ns_str
1141 \str_new:N \l_stex_modules_subpath_str

```

```

1142 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1143   \str_set:Nx \l_tmpa_str { #1 }
1144   \seq_set_eq:NN \l_tmpa_seq #2
1145   % split off file extension
1146   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1147   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1148   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1149   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1150
1151   \bool_set_true:N \l_tmpa_bool
1152   \bool_while_do:Nn \l_tmpa_bool {
1153     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1154     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1155       {source} { \bool_set_false:N \l_tmpa_bool }
1156     }{}{
1157       \seq_if_empty:NT \l_tmpa_seq {
1158         \bool_set_false:N \l_tmpa_bool
1159       }
1160     }
1161   }
1162
1163   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1164   \str_if_empty:NTF \l_stex_modules_subpath_str {
1165     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1166   }{
1167     \str_set:Nx \l_stex_modules_ns_str {
1168       \l_tmpa_str/\l_stex_modules_subpath_str
1169     }
1170   }
1171 }
1172
1173 \cs_new_protected:Nn \stex_modules_current_namespace: {
1174   \str_clear:N \l_stex_modules_subpath_str
1175   \prop_if_exist:NTF \l_stex_current_repository_prop {
1176     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1177     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1178   }{
1179     % split off file extension
1180     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1181     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1182     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1183     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1184     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1185     \str_set:Nx \l_stex_modules_ns_str {
1186       file:\stex_path_to_string:N \l_tmpa_seq
1187     }
1188   }
1189 }

```

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

28.1 The smodule environment

smodule arguments:

```

1190 \keys_define:nn { stex / module } {
1191   title      .tl_set:N      = \smoduletitle ,
1192   type       .str_set_x:N   = \smoduletype ,
1193   id         .str_set_x:N   = \smoduleid ,
1194   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1195   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1196   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1197   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1198   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1199   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1200   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1201   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1202 }
1203
1204 \cs_new_protected:Nn \__stex_modules_args:n {
1205   \str_clear:N \smoduletitle
1206   \str_clear:N \smoduletype
1207   \str_clear:N \smoduleid
1208   \str_clear:N \l_stex_module_ns_str
1209   \str_clear:N \l_stex_module_deprecate_str
1210   \str_clear:N \l_stex_module_lang_str
1211   \str_clear:N \l_stex_module_sig_str
1212   \str_clear:N \l_stex_module_creators_str
1213   \str_clear:N \l_stex_module_contributors_str
1214   \str_clear:N \l_stex_module_meta_str
1215   \str_clear:N \l_stex_module_srccite_str
1216   \keys_set:nn { stex / module } { #1 }
1217 }
1218
1219 % module parameters here? In the body?
1220

```

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

```

1221 \cs_new_protected:Nn \stex_module_setup:nn {
1222   \str_set:Nx \l_stex_module_name_str { #2 }
1223   \__stex_modules_args:n { #1 }
1224
1225   First, we set up the name and namespace of the module.
1226   Are we in a nested module?
1227
1228   \stex_if_in_module:TF {
1229     % Nested module
1230     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1231     { ns } \l_stex_module_ns_str
1232     \str_set:Nx \l_stex_module_name_str {
1233       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1234       { name } / \l_stex_module_name_str
1235     }
1236   }{
1237     % not nested:
1238     \str_if_empty:NT \l_stex_module_ns_str {
1239       \stex_modules_current_namespace:
1240     }
1241   }
1242 }

```

```

1236 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1237 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1238 / {\l_stex_module_ns_str}
1239 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1240 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1241 \str_set:Nx \l_stex_module_ns_str {
1242 \stex_path_to_string:N \l_tmpa_seq
1243 }
1244 }
1245 }
1246 }

```

Next, we determine the language of the module:

```

1247 \str_if_empty:NT \l_stex_module_lang_str {
1248 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1249 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1250 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1251 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1252 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1253 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1254 inferred~from~file~name}
1255 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1256 }
1257 }
1258
1259 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1260 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1261 \l_tmpa_str {
1262 \ltx@ifpackageloaded{babel}{
1263 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1264 }{}
1265 } {
1266 \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
1267 }
1268 }}

```

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

```

1269 \str_if_empty:NTF \l_stex_module_sig_str {
1270 \exp_args:Nnx \prop_gset_from_keyval:cn {
1271 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1272 } {
1273 name = \l_stex_module_name_str ,
1274 ns = \l_stex_module_ns_str ,
1275 file = \exp_not:o { \g_stex_currentfile_seq } ,
1276 lang = \l_stex_module_lang_str ,
1277 sig = \l_stex_module_sig_str ,
1278 deprecate = \l_stex_module_deprecate_str ,
1279 meta = \l_stex_module_meta_str
1280 }
1281 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1282 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1283 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1284 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1285 \str_if_empty:NT \l_stex_module_meta_str {
1286   \str_set:Nx \l_stex_module_meta_str {
1287     \c_stex_metatheory_ns_str ? Metatheory
1288   }
1289 }
1290 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1291   \bool_set_true:N \l_stex_in_meta_bool
1292   \exp_args:Nx \stex_add_to_current_module:n {
1293     \bool_set_true:N \l_stex_in_meta_bool
1294     \stex_activate_module:n {\l_stex_module_meta_str}
1295     \bool_set_false:N \l_stex_in_meta_bool
1296   }
1297   \stex_activate_module:n {\l_stex_module_meta_str}
1298   \bool_set_false:N \l_stex_in_meta_bool
1299 }
1300 }{
1301   \str_if_empty:NT \l_stex_module_lang_str {
1302     \msg_error:nnxx{stex}{error/siglanguage}{
1303       \l_stex_module_ns_str?\l_stex_module_name_str
1304     }\l_stex_module_sig_str}
1305   }
1306
1307   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1308   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1309   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1310   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1311   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1312   \str_set:Nx \l_tmpa_str {
1313     \stex_path_to_string:N \l_tmpa_seq /
1314     \l_tmpa_str . \l_stex_module_sig_str .tex
1315   }
1316   \IfFileExists \l_tmpa_str {
1317     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1318       \str_clear:N \l_stex_current_module_str
1319       \seq_clear:N \l_stex_all_modules_seq
1320       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1321     }
1322   }{
1323     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1324   }
1325   \stex_if_smsmode:F {
1326     \stex_activate_module:n {
1327       \l_stex_module_ns_str ? \l_stex_module_name_str
1328     }
1329   }
1330   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1331 }
1332 \str_if_empty:NF \l_stex_module_deprecate_str {
1333   \msg_warning:nnxx{stex}{warning/deprecated}{
1334     Module~\l_stex_current_module_str
1335   }{
1336     \l_stex_module_deprecate_str
1337   }

```

```

1338 }
1339 }

```

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

smodule The module environment.

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

```

1340 \cs_new_protected:Nn \_stex_modules_begin_module: {
1341   \stex_reactivate_macro:N \STEXexport
1342   \stex_reactivate_macro:N \importmodule
1343   \stex_reactivate_macro:N \symdecl
1344   \stex_reactivate_macro:N \notation
1345   \stex_reactivate_macro:N \symdef
1346
1347   \stex_debug:nn{modules}{
1348     New~module:\\
1349     Namespace:~\l_stex_module_ns_str\\
1350     Name:~\l_stex_module_name_str\\
1351     Language:~\l_stex_module_lang_str\\
1352     Signature:~\l_stex_module_sig_str\\
1353     Metatheory:~\l_stex_module_meta_str\\
1354     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1355   }
1356
1357   \seq_put_right:Nx \l_stex_all_modules_seq {
1358     \l_stex_module_ns_str ? \l_stex_module_name_str
1359   }
1360
1361   \stex_if_smsmode:F{
1362     \begin{stex_annotate_env} {theory} {
1363       \l_stex_module_ns_str ? \l_stex_module_name_str
1364     }
1365
1366     \stex_annotate_invisible:nnn{header}{} {
1367       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1368       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1369       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1370         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1371       }
1372       \str_if_empty:NF \smoduletype {
1373         \stex_annotate:nnn{type}{\smoduletype}{}
1374       }
1375     }
1376   }
1377   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1378   % TODO: Inherit metatheory for nested modules?
1379 }
1380 \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}`

```

1381 \cs_new_protected:Nn \_stex_modules_end_module: {

```



```

1382 \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module
1383 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1384 \iffalse \begin{stex_annotate_env} \fi %%^A make syntax highlighting work again
1385 \NewDocumentEnvironment { smodule } { 0{} m } {
1386   \stex_module_setup:nn{#1}{#2}
1387   \par
1388   \stex_if_smsmode:F{
1389     \tl_clear:N \l_tmpa_tl
1390     \clist_map_inline:Nn \smodulotype {
1391       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1392         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1393       }
1394     }
1395     \tl_if_empty:NTF \l_tmpa_tl {
1396       \__stex_modules_smodule_start:
1397     }{
1398       \l_tmpa_tl
1399     }
1400   }
1401   \__stex_modules_begin_module:
1402   \str_if_empty:NF \smoduleid {
1403     \stex_ref_new_doc_target:n \smoduleid
1404   }
1405   \stex_smsmode_do:
1406 } {
1407   \__stex_modules_end_module:
1408   \stex_if_smsmode:F {
1409     \end{stex_annotate_env}
1410     \clist_set:Nn \l_tmpa_clist \smodulotype
1411     \tl_clear:N \l_tmpa_tl
1412     \clist_map_inline:Nn \l_tmpa_clist {
1413       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1414         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1415       }
1416     }
1417     \tl_if_empty:NTF \l_tmpa_tl {
1418       \__stex_modules_smodule_end:
1419     }{
1420       \l_tmpa_tl
1421     }
1422   }
1423 }

```

\stexpatchmodule

```

1424 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1425 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1426
1427 \newcommand\stexpatchmodule[3] [] {
1428   \str_set:Nx \l_tmpa_str{ #1 }
1429   \str_if_empty:NTF \l_tmpa_str {
1430     \tl_set:Nn \__stex_modules_smodule_start: { #2 }

```

```

1431     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1432   }{
1433     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1434     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1435   }
1436 }

```

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

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1437 \NewDocumentCommand \STEXModule { m } {
1438   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1439   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1440   \tl_set:Nn \l_tmpa_tl {
1441     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1442   }
1443   \seq_map_inline:Nn \l_stex_all_modules_seq {
1444     \str_set:Nn \l_tmpb_str { ##1 }
1445     \str_if_eq:eeT { \l_tmpa_str } {
1446       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1447     } {
1448       \seq_map_break:n {
1449         \tl_set:Nn \l_tmpa_tl {
1450           \stex_invoke_module:n { ##1 }
1451         }
1452       }
1453     }
1454   }
1455   \l_tmpa_tl
1456 }
1457
1458 \cs_new_protected:Nn \stex_invoke_module:n {
1459   \stex_debug:nn{modules}{Invoking~module~#1}
1460   \peek_charcode_remove:NTF ! {
1461     \__stex_modules_invoke_uri:nN { #1 }
1462   } {
1463     \peek_charcode_remove:NTF ? {
1464       \__stex_modules_invoke_symbol:nn { #1 }
1465     } {
1466       \msg_error:nnx{stex}{error/syntax}{
1467         ?~or~!~expected~after~
1468         \c_backslash_str STEXModule{#1}
1469       }
1470     }
1471   }
1472 }
1473
1474 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1475   \str_set:Nn #2 { #1 }
1476 }
1477

```

```

1478 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1479   \stex_invoke_symbol:n{#1?#2}
1480 }

```

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

\stex_activate_module:n

```

1481 \bool_new:N \l_stex_in_meta_bool
1482 \bool_set_false:N \l_stex_in_meta_bool
1483 \cs_new_protected:Nn \stex_activate_module:n {
1484   \stex_debug:nn{modules}{Activating~module~#1}
1485   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1486     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1487   }
1488   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1489     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1490     \use:c{ c_stex_module_#1_code }
1491   }
1492 }

```

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

```

1493 \</package>

```

Chapter 29

STEX -Module Inheritance Implementation

```
1494 <*package>
1495
1496 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1497
```

29.1 SMS Mode

```
1498 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1499 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1500 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1501 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1502
1503 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1504   \makeatletter
1505   \makeatother
1506   \ExplSyntaxOn
1507   \ExplSyntaxOff
1508   \rustexBREAK
1509 }
1510
1511 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1512   \symdef
1513   \importmodule
1514   \notation
1515   \symdecl
1516   \STEXexport
1517   \inlineass
1518   \inlinedef
1519   \inlineex
1520   \endinput
1521   \setnotation
```

```

1522 \copynotation
1523 }
1524
1525 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1526   \tl_to_str:n {
1527     smodule,
1528     copymodule,
1529     interpretmodule,
1530     sdefinition,
1531     sexample,
1532     sassertion,
1533     sparagraph
1534   }
1535 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 31.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1536 \bool_new:N \g__stex_smsmode_bool
1537 \bool_set_false:N \g__stex_smsmode_bool
1538 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1539   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1540 }

```

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

`\stex_in_smsmode:nn`

```

1541 \cs_new_protected:Nn \stex_in_smsmode:nn {
1542   \vbox_set:Nn \l_tmpa_box {
1543     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1544     \bool_gset_true:N \g__stex_smsmode_bool
1545     #2
1546     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1547   }
1548   \box_clear:N \l_tmpa_box
1549 }
1550
1551 \quark_new:N \q__stex_smsmode_break
1552
1553 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1554   \stex_filestack_push:n{#1}
1555   \stex_in_smsmode:nn{#1} {
1556     #2
1557     \everyeof{\q__stex_smsmode_break\noexpand}
1558     \expandafter\expandafter\expandafter
1559     \stex_smsmode_do:
1560     \csname @ @ input\endcsname "#1"\relax
1561   }
1562   \stex_filestack_pop:
1563 }

```

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

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

```

1564 \cs_new_protected:Npn \stex_smsmode_do: {
1565   \stex_if_smsmode:T {
1566     \__stex_smsmode_do:w
1567   }
1568 }
1569 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1570   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1571     \expandafter\if\expandafter\relax\noexpand#1
1572     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1573   } \else\expandafter\__stex_smsmode_do:w\fi
1574 }{
1575   \__stex_smsmode_do:w % #1
1576 }
1577 }
1578 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1579   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1580     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1581       #1\__stex_smsmode_do:w
1582     }{
1583       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1584         #1
1585       }{
1586         \cs_if_eq:NNTF \begin #1 {
1587           \__stex_smsmode_check_begin:n
1588         }{
1589           \cs_if_eq:NNTF \end #1 {
1590             \__stex_smsmode_check_end:n
1591           }{
1592             \__stex_smsmode_do:w
1593           }
1594         }
1595       }
1596     }
1597   }
1598 }
1599
1600 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1601   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1602     \begin{#1}
1603   }{
1604     \__stex_smsmode_do:w
1605   }
1606 }
1607 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1608   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1609     \end{#1}\__stex_smsmode_do:w
1610   }{
1611     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1612   }
1613 }

```

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

29.2 Inheritance

1614 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1615 \cs_new_protected:Nn \stex_import_module_uri:nn {
1616   \str_set:Nx \l_stex_import_archive_str { #1 }
1617   \str_set:Nn \l_stex_import_path_str { #2 }
1618
1619   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1620   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1621   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1622
1623   \stex_modules_current_namespace:
1624   \bool_lazy_all:nTF {
1625     {\str_if_empty_p:N \l_stex_import_archive_str}
1626     {\str_if_empty_p:N \l_stex_import_path_str}
1627     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1628   }{
1629     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1630     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1631   }{
1632     \str_if_empty:NT \l_stex_import_archive_str {
1633       \prop_if_exist:NT \l_stex_current_repository_prop {
1634         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1635       }
1636     }
1637     \str_if_empty:NTF \l_stex_import_archive_str {
1638       \str_if_empty:NF \l_stex_import_path_str {
1639         \str_set:Nx \l_stex_import_ns_str {
1640           \l_stex_module_ns_str / \l_stex_import_path_str
1641         }
1642       }
1643     }{
1644       \stex_require_repository:n \l_stex_import_archive_str
1645       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1646       \l_stex_import_ns_str
1647       \str_if_empty:NF \l_stex_import_path_str {
1648         \str_set:Nx \l_stex_import_ns_str {
1649           \l_stex_import_ns_str / \l_stex_import_path_str
1650         }
1651       }
1652     }
1653   }
1654 }
```

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

<code>\l_stex_import_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_import_archive_str</code>	1655 <code>\str_new:N \l_stex_import_name_str</code>
<code>\l_stex_import_path_str</code>	1656 <code>\str_new:N \l_stex_import_archive_str</code>
<code>\l_stex_import_ns_str</code>	1657 <code>\str_new:N \l_stex_import_path_str</code>
	1658 <code>\str_new:N \l_stex_import_ns_str</code>

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

```

\stex_import_require_module:nnnn      {\<ns>} {\<archive-ID>} {\<path>} {\<name>}
1659 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1660   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1661
1662     % archive
1663     \str_set:Nx \l_tmpa_str { #2 }
1664     \str_if_empty:NTF \l_tmpa_str {
1665       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1666     } {
1667       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1668       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1669       \seq_put_right:Nn \l_tmpa_seq { source }
1670     }
1671
1672     % path
1673     \str_set:Nx \l_tmpb_str { #3 }
1674     \str_if_empty:NTF \l_tmpb_str {
1675       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1676
1677       \ltx@ifpackageloaded{babel} {
1678         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1679           { \language } \l_tmpb_str {
1680           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1681         }
1682       } {
1683         \str_clear:N \l_tmpb_str
1684       }
1685
1686       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1687       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1688         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1689       }{
1690         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1691         \IfFileExists{ \l_tmpa_str.tex }{
1692           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1693         }{
1694           % try english as default
1695           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1696           \IfFileExists{ \l_tmpa_str.en.tex }{
1697             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1698           }{
1699             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1700           }
1701         }
1702       }
1703
1704     } {
1705       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1706       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1707
1708       \ltx@ifpackageloaded{babel} {
1709         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1710           { \language } \l_tmpb_str {
1711           \msg_error:nnx{stex}{error/unknownlanguage}{\language}

```



```

1712     }
1713 } {
1714     \str_clear:N \l_tmpb_str
1715 }
1716
1717 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1718
1719 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1720 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1721     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1722 }{
1723     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1724     \IfFileExists{ \l_tmpa_str/#4.tex }{
1725         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1726     }{
1727         % try english as default
1728         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1729         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1730             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1731         }{
1732             \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1733             \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1734                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1735             }{
1736                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1737                 \IfFileExists{ \l_tmpa_str.tex }{
1738                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1739                 }{
1740                     % try english as default
1741                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1742                     \IfFileExists{ \l_tmpa_str.en.tex }{
1743                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1744                     }{
1745                         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1746                     }
1747                 }
1748             }
1749         }
1750     }
1751 }
1752 }
1753
1754 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1755     \seq_clear:N \l_stex_all_modules_seq
1756     \str_clear:N \l_stex_current_module_str
1757     \str_set:Nx \l_tmpb_str { #2 }
1758     \str_if_empty:NF \l_tmpb_str {
1759         \stex_set_current_repository:n { #2 }
1760     }
1761     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1762 }
1763
1764 \stex_if_module_exists:nF { #1 ? #4 } {
1765     \msg_error:nnx{stex}{error/unknownmodule}{

```

```

1766         #1?#4~(in~file~\g__stex_importmodule_file_str)
1767     }
1768 }
1769 }
1770 \stex_activate_module:n { #1 ? #4 }
1771 }

```

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

`\importmodule`

```

1772 \NewDocumentCommand \importmodule { 0{} m } {
1773   \stex_import_module_uri:nn { #1 } { #2 }
1774   \stex_debug:nn{modules}{Importing~module:~
1775     \l_stex_import_ns_str ? \l_stex_import_name_str
1776   }
1777   \stex_if_smsmode:F {
1778     \stex_import_require_module:nnnn
1779     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1780     { \l_stex_import_path_str } { \l_stex_import_name_str }
1781     \stex_annotate_invisible:nnn
1782     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1783   }
1784   \exp_args:Nx \stex_add_to_current_module:n {
1785     \stex_import_require_module:nnnn
1786     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1787     { \l_stex_import_path_str } { \l_stex_import_name_str }
1788   }
1789   \exp_args:Nx \stex_add_import_to_current_module:n {
1790     \l_stex_import_ns_str ? \l_stex_import_name_str
1791   }
1792   \stex_smsmode_do:
1793   \ignorespacesandpars
1794 }
1795 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1796 \NewDocumentCommand \usemodule { 0{} m } {
1797   \stex_if_smsmode:F {
1798     \stex_import_module_uri:nn { #1 } { #2 }
1799     \stex_import_require_module:nnnn
1800     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1801     { \l_stex_import_path_str } { \l_stex_import_name_str }
1802     \stex_annotate_invisible:nnn
1803     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1804   }
1805   \stex_smsmode_do:
1806   \ignorespacesandpars
1807 }

```

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

```

1808 </package>

```

Chapter 30

STEX -Symbols Implementation

```
1809 <*package>
1810
1811 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1812
1813 Warnings and error messages
1814 \msg_new:nnn{stex}{error/wrongargs}{
1815   args~value~in~symbol~declaration~for~#1~
1816   needs~to~be~i,~a,~b~or~B,~but~#2~given
1817 }
```

30.1 Symbol Declarations

```
1817 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1818 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)
```

\STEXsymbol

```
1819 \NewDocumentCommand \STEXsymbol { m } {
1820   \stex_get_symbol:n { #1 }
1821   \exp_args:No
1822   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1823 }
```

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

symdecl arguments:

```
1824 \keys_define:nn { stex / symdecl } {
1825   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1826   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1827   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1828   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1829   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1830   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
```

```

1831 gfc .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1832 specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1833 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1834 assoc .choices:nn =
1835 {bin,binl,binr,pre,conj,pwconj}
1836 {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1837 }
1838
1839 \bool_new:N \l_stex_symdecl_make_macro_bool
1840
1841 \cs_new_protected:Nn \__stex_symdecl_args:n {
1842 \str_clear:N \l_stex_symdecl_name_str
1843 \str_clear:N \l_stex_symdecl_args_str
1844 \str_clear:N \l_stex_symdecl_deprecate_str
1845 \str_clear:N \l_stex_symdecl_astype_str
1846 \bool_set_false:N \l_stex_symdecl_local_bool
1847 \tl_clear:N \l_stex_symdecl_type_tl
1848 \tl_clear:N \l_stex_symdecl_definiens_tl
1849
1850 \keys_set:nn { stex / symdecl } { #1 }
1851 }

```

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

```

1852
1853 \NewDocumentCommand \symdecl { s O{} m } {
1854 \__stex_symdecl_args:n { #2 }
1855 \IfBooleanTF #1 {
1856 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1857 } {
1858 \bool_set_true:N \l_stex_symdecl_make_macro_bool
1859 }
1860 \stex_symdecl_do:n { #3 }
1861 \stex_smsmode_do:
1862 }
1863
1864 \cs_new_protected:Nn \stex_symdecl_do:nn {
1865 \__stex_symdecl_args:n{#1}
1866 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1867 \stex_symdecl_do:n{#2}
1868 }
1869
1870 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1871 \cs_new_protected:Nn \stex_symdecl_do:n {
1872 \stex_if_in_module:F {
1873 % TODO throw error? some default namespace?
1874 }
1875
1876 \str_if_empty:NT \l_stex_symdecl_name_str {
1877 \str_set:Nx \l_stex_symdecl_name_str { #1 }

```

```

1878 }
1879
1880 \prop_if_exist:cT { \l_stex_symdecl_
1881   \l_stex_current_module_str ?
1882   \l_stex_symdecl_name_str
1883   _prop
1884 }{
1885   % TODO throw error (beware of circular dependencies)
1886 }
1887
1888 \prop_clear:N \l_tmpa_prop
1889 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1890 \seq_clear:N \l_tmpa_seq
1891 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1892 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1893
1894 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1895   \str_if_empty:NF \l_stex_module_deprecate_str {
1896     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1897   }
1898 }
1899 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1900
1901 \exp_args:No \stex_add_constant_to_current_module:n {
1902   \l_stex_symdecl_name_str
1903 }
1904
1905 % arity/args
1906 \int_zero:N \l_tmpb_int
1907
1908 \bool_set_true:N \l_tmpa_bool
1909 \str_map_inline:Nn \l_stex_symdecl_args_str {
1910   \token_case_meaning:NnF ##1 {
1911     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1912     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1913     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1914     {\tl_to_str:n a} {
1915       \bool_set_false:N \l_tmpa_bool
1916       \int_incr:N \l_tmpb_int
1917     }
1918     {\tl_to_str:n B} {
1919       \bool_set_false:N \l_tmpa_bool
1920       \int_incr:N \l_tmpb_int
1921     }
1922   }{
1923     \msg_error:nnxx{stex}{error/wrongargs}{
1924       \l_stex_current_module_str ?
1925       \l_stex_symdecl_name_str
1926     }{##1}
1927   }
1928 }
1929 \bool_if:NTF \l_tmpa_bool {
1930   % possibly numeric
1931   \str_if_empty:NTF \l_stex_symdecl_args_str {

```

```

1932     \prop_put:Nnn \l_tmpa_prop { args } {}
1933     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1934   }{
1935     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1936     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1937     \str_clear:N \l_tmpa_str
1938     \int_step_inline:nn \l_tmpa_int {
1939       \str_put_right:Nn \l_tmpa_str i
1940     }
1941     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1942   }
1943 } {
1944   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1945   \prop_put:Nnx \l_tmpa_prop { arity }
1946     { \str_count:N \l_stex_symdecl_args_str }
1947 }
1948 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1949
1950
1951 % semantic macro
1952
1953 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1954   \exp_args:Nx \stex_do_up_to_module:n {
1955     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1956       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1957     }}
1958   }
1959
1960   \bool_if:NF \l_stex_symdecl_local_bool {
1961     \exp_args:Nx \stex_add_to_current_module:n {
1962       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1963         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1964       } }
1965     }
1966   }
1967 }
1968
1969 % add to all symbols
1970
1971 \bool_if:NF \l_stex_symdecl_local_bool {
1972   \exp_args:Nx \stex_add_to_current_module:n {
1973     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1974       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1975     }
1976   }
1977   % \exp_args:Nx \stex_add_field_to_current_module:n {
1978   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979   % }
1980 }
1981
1982 \stex_debug:nn{symbols}{New~symbol:~
1983   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1984   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1985   Args:~\prop_item:Nn \l_tmpa_prop { args }

```

```

1986 }
1987
1988 % circular dependencies require this:
1989
1990 \prop_if_exist:cF {
1991   l_stex_symdecl_
1992   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1993   _prop
1994 } {
1995   \prop_set_eq:cN {
1996     l_stex_symdecl_
1997     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1998     _prop
1999   } \l_tmpa_prop
2000 }
2001
2002 \seq_clear:c {
2003   l_stex_symdecl_
2004   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2005   _notations
2006 }
2007
2008 \bool_if:NF \l_stex_symdecl_local_bool {
2009   \exp_args:Nx
2010   \stex_add_to_current_module:n {
2011     \seq_clear:c {
2012       l_stex_symdecl_
2013       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2014       _notations
2015     }
2016     \prop_set_from_keyval:cn {
2017       l_stex_symdecl_
2018       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2019       _prop
2020     } {
2021       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2022       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2023       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2024       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2025       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2026       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2027     }
2028   }
2029 }
2030
2031 \stex_if_smsmode:F {
2032   \exp_args:Nx \stex_do_up_to_module:n {
2033     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2034       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2035     }
2036   }
2037   \stex_if_do_html:T {
2038     \stex_annotate_invisible:nnn {symdecl} {
2039       \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2040 } {
2041   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st
2042   \stex_annotate_invisible:nnn{args}{}}{
2043     \prop_item:Nn \l_tmpa_prop { args }
2044   }
2045   \stex_annotate_invisible:nnn{macroname}{#1}{}
2046   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2047     \stex_annotate_invisible:nnn{definiens}{}
2048     {${\l_stex_symdecl_definiens_tl$}
2049   }
2050   \str_if_empty:NF \l_stex_symdecl_assocotype_str {
2051     \stex_annotate_invisible:nnn{assocotype}{\l_stex_symdecl_assocotype_str}{}
2052   }
2053 }
2054 }
2055 }
2056 }

```

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

`\stex_get_symbol:n`

```

2057 \str_new:N \l_stex_get_symbol_uri_str
2058
2059 \cs_new_protected:Nn \stex_get_symbol:n {
2060   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2061     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2062   }{
2063     % argument is a string
2064     % is it a command name?
2065     \cs_if_exist:cTF { #1 }{
2066       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2067       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2068       \str_if_empty:NNTF \l_tmpa_str {
2069         \exp_args:Nx \cs_if_eq:NNTF {
2070           \tl_head:N \l_tmpa_tl
2071         } \stex_invoke_symbol:n {
2072           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2073         }{
2074           \__stex_symdecl_get_symbol_from_string:n { #1 }
2075         }
2076       } {
2077         \__stex_symdecl_get_symbol_from_string:n { #1 }
2078       }
2079     }{
2080       % argument is not a command name
2081       \__stex_symdecl_get_symbol_from_string:n { #1 }
2082       % \l_stex_all_symbols_seq
2083     }
2084   }
2085   \str_if_eq:eeF {
2086     \prop_item:cn {
2087       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2088     }{ deprecate }
2089   }{}{

```



```

2090 \msg_warning:nnxx{stex}{warning/deprecated}{
2091   Symbol~\l_stex_get_symbol_uri_str
2092 }{
2093   \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str _prop}{ deprecate }
2094 }
2095 }
2096 }
2097
2098 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2099   \str_set:Nn \l_tmpa_str { #1 }
2100   \bool_set_false:N \l_tmpa_bool
2101   \stex_if_in_module:T {
2102     \exp_args:Nno \seq_if_in:cnT {c_stex_module_l_stex_current_module_str _constants} { \l_
2103       \bool_set_true:N \l_tmpa_bool
2104       \str_set:Nx \l_stex_get_symbol_uri_str {
2105         \l_stex_current_module_str ? #1
2106       }
2107     }
2108   }
2109   \bool_if:NF \l_tmpa_bool {
2110     \tl_set:Nn \l_tmpa_tl {
2111       \msg_set:nnn{stex}{error/unknownsymbol}{
2112         No~symbol~#1~found!
2113       }
2114       \msg_error:nn{stex}{error/unknownsymbol}
2115     }
2116     \str_set:Nn \l_tmpa_str { #1 }
2117     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2118     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2119       \str_set:Nn \l_tmpb_str { ##1 }
2120       \str_if_eq:eeT { \l_tmpa_str } {
2121         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2122       } {
2123         \seq_map_break:n {
2124           \tl_set:Nn \l_tmpa_tl {
2125             \str_set:Nn \l_stex_get_symbol_uri_str {
2126               ##1
2127             }
2128           }
2129         }
2130       }
2131     }
2132     \l_tmpa_tl
2133   }
2134 }
2135
2136 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2137   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2138     { \tl_tail:N \l_tmpa_tl }
2139   \tl_if_single:NTF \l_tmpa_tl {
2140     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2141       \exp_after:wN \str_set:Nn \exp_after:wN
2142         \l_stex_get_symbol_uri_str \l_tmpa_tl
2143     }{

```

```

2144     % TODO
2145     % tail is not a single group
2146   }
2147 }{
2148   % TODO
2149   % tail is not a single group
2150 }
2151 }

```

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

30.2 Notations

```

2152 <@@=stex_notation>

notation arguments:
2153 \keys_define:nn { stex / notation } {
2154   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2155   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2156   prec .str_set_x:N = \l__stex_notation_prec_str ,
2157   op .tl_set:N = \l__stex_notation_op_tl ,
2158   primary .bool_set:N = \l__stex_notation_primary_bool ,
2159   primary .default:n = {true} ,
2160   unknown .code:n = \str_set:Nx
2161     \l__stex_notation_variant_str \l_keys_key_str
2162 }
2163
2164 \cs_new_protected:Nn \stex_notation_args:n {
2165   \str_clear:N \l__stex_notation_lang_str
2166   \str_clear:N \l__stex_notation_variant_str
2167   \str_clear:N \l__stex_notation_prec_str
2168   \tl_clear:N \l__stex_notation_op_tl
2169   \bool_set_false:N \l__stex_notation_primary_bool
2170
2171   \keys_set:nn { stex / notation } { #1 }
2172 }

```

\notation

```

2173 \NewDocumentCommand \notation { s O{} m } {
2174   \stex_notation_args:n { #2 }
2175   \tl_clear:N \l_stex_symdecl_definiens_tl
2176   \stex_get_symbol:n { #3 }
2177   \tl_set:Nn \l__stex_notation_after_do_tl {
2178     \__stex_notation_final:
2179     \IfBooleanTF#1{
2180       \stex_setnotation:n {\l__stex_notation_symbol_str}
2181     }{}
2182     \stex_smsmode_do:
2183   }
2184   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2185 }
2186 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

`\stex_notation_do:nn`

```

2187 \seq_new:N \l__stex_notation_precedences_seq
2188 \tl_new:N \l__stex_notation_opprec_tl
2189 \int_new:N \l__stex_notation_currarg_int
2190
2191 \cs_new_protected:Nn \stex_notation_do:nn {
2192   \let\l_stex_current_symbol_str\relax
2193   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2194   \seq_clear:N \l__stex_notation_precedences_seq
2195   \tl_clear:N \l__stex_notation_opprec_tl
2196   \prop_get:cnN {
2197     l_stex_symdecl_ #1 _prop
2198   } { args } \l__stex_notation_args_str
2199
2200   % precedences
2201   \prop_get:cnN {
2202     l_stex_symdecl_ #1 _prop
2203   } { arity } \l__stex_notation_arity_str
2204   \str_if_empty:NTF \l__stex_notation_prec_str {
2205     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2206       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2207     }{
2208       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2209     }
2210   } {
2211     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2212       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2213       \int_step_inline:nn { \l__stex_notation_arity_str } {
2214         \exp_args:NNo
2215         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2216       }
2217     }{
2218       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2219       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2220         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2221         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2222           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2223             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2224           \seq_map_inline:Nn \l_tmpa_seq {
2225             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2226           }
2227         }
2228       }{
2229         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2230           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2231         }{
2232           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2233         }
2234       }
2235     }
2236   }
2237
2238   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2239   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2240 \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2241 \exp_args:NNo
2242 \seq_put_right:No \l__stex_notation_precedences_seq {
2243 \l__stex_notation_opprec_tl
2244 }
2245 }
2246 }
2247
2248 \tl_clear:N \l__stex_notation_dummyargs_tl
2249
2250 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2251 \exp_args:NNe
2252 \cs_set:Npn \l__stex_notation_macrocode_cs {
2253 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2254 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2255 { \l__stex_notation_opprec_tl }
2256 { \exp_not:n { #2 } }
2257 }
2258 \l__stex_notation_after_do_tl
2259 }{
2260 \str_if_in:NnTF \l__stex_notation_args_str b {
2261 \exp_args:Nne \use:nn
2262 {
2263 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2264 \cs_set:Npn \l__stex_notation_arity_str } { {
2265 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2266 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2267 { \l__stex_notation_opprec_tl }
2268 { \exp_not:n { #2 } }
2269 }}
2270 }{
2271 \str_if_in:NnTF \l__stex_notation_args_str B {
2272 \exp_args:Nne \use:nn
2273 {
2274 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2275 \cs_set:Npn \l__stex_notation_arity_str } { {
2276 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2277 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2278 { \l__stex_notation_opprec_tl }
2279 { \exp_not:n { #2 } }
2280 } }
2281 }{
2282 \exp_args:Nne \use:nn
2283 {
2284 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2285 \cs_set:Npn \l__stex_notation_arity_str } { {
2286 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2287 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2288 { \l__stex_notation_opprec_tl }
2289 { \exp_not:n { #2 } }
2290 } }
2291 }
2292 }
2293

```

```

2294 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2295 \int_zero:N \l__stex_notation_currarg_int
2296 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2297 \__stex_notation_arguments:
2298 }
2299 }

```

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

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

```

2300 \cs_new_protected:Nn \__stex_notation_arguments: {
2301 \int_incr:N \l__stex_notation_currarg_int
2302 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2303 \l__stex_notation_after_do_tl
2304 }{
2305 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2306 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2307 \str_if_eq:VnTF \l_tmpa_str a {
2308 \__stex_notation_argument_assoc:n
2309 }{
2310 \str_if_eq:VnTF \l_tmpa_str B {
2311 \__stex_notation_argument_assoc:n
2312 }{
2313 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2314 \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2315 { \stex_term_math_arg:nnn
2316 { \int_use:N \l__stex_notation_currarg_int }
2317 { \l_tmpa_str }
2318 { ###\int_use:N \l__stex_notation_currarg_int }
2319 }
2320 }
2321 \__stex_notation_arguments:
2322 }
2323 }
2324 }
2325 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2326 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2327
2328 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2329 {\l__stex_notation_arity_str}{
2330 #1
2331 }
2332 \int_zero:N \l_tmpa_int
2333 \tl_clear:N \l_tmpa_tl
2334 \str_map_inline:Nn \l__stex_notation_args_str {
2335 \int_incr:N \l_tmpa_int
2336 \tl_put_right:Nx \l_tmpa_tl {
2337 \str_if_eq:nnTF {##1}{a}{ {} }{
2338 \str_if_eq:nnTF {##1}{B}{ {} }{
2339 {##### \int_use:N \l_tmpa_int}

```

```

2340     }
2341   }
2342 }
2343 }
2344 \exp_after:wN\exp_after:wN\exp_after:wN \def
2345 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2346 \exp_after:wN\exp_after:wN\exp_after:wN ##
2347 \exp_after:wN\exp_after:wN\exp_after:wN 1
2348 \exp_after:wN\exp_after:wN\exp_after:wN ##
2349 \exp_after:wN\exp_after:wN\exp_after:wN 2
2350 \exp_after:wN\exp_after:wN\exp_after:wN {
2351   \exp_after:wN \exp_after:wN \exp_after:wN
2352   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2353     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2354   }
2355 }
2356
2357 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2358 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2359   \stex_term_math_assoc_arg:nnnn
2360   { \int_use:N \l__stex_notation_currarg_int }
2361   { \l_tmpa_str }
2362   { ####\int_use:N \l__stex_notation_currarg_int }
2363   { \l_tmpa_cs {####1} {####2} }
2364 } }
2365 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2366 %\tl_put_right:Nx \l_tmpa_tl {
2367 % { \stex_term_math_assoc_arg:nnnn
2368 %   { \int_use:N \l_tmpa_int }
2369 %   { \l_tmpb_str }
2370 %   \exp_args:No \exp_not:n
2371 %   {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2372 %   { ####\int_use:N \l_tmpa_int }
2373 % }
2374 %}
2375 \__stex_notation_arguments:
2376 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2377 \cs_new_protected:Nn \__stex_notation_final: {
2378   \exp_args:Nne \use:nn
2379   {
2380     \cs_generate_from_arg_count:cNnn {
2381       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2382       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2383       _cs
2384     }
2385     \cs_set:Npn \l__stex_notation_arity_str } { {
2386       \exp_after:wN \exp_after:wN \exp_after:wN
2387       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2388       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2389     } }

```

```

2390
2391 \tl_if_empty:NF \l__stex_notation_op_tl {
2392   \cs_set:cpx {
2393     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2394     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2395     _cs
2396   } {
2397     \stex_term_oms:nnn {
2398       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2399       \l__stex_notation_lang_str
2400     }{
2401       \l__stex_notation_symbol_str
2402     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2403   }
2404 }
2405
2406 \exp_args:Ne
2407 \stex_add_to_current_module:n {
2408   \cs_generate_from_arg_count:cNnn {
2409     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2410     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2411     _cs
2412   } \cs_set:Npn {\l__stex_notation_arity_str} {
2413     \exp_after:wN \exp_after:wN \exp_after:wN
2414     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2415     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2416   }
2417   \tl_if_empty:NF \l__stex_notation_op_tl {
2418     \cs_set:cpn {
2419       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2420       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2421       _cs
2422     } {
2423       \stex_term_oms:nnn {
2424         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2425         \l__stex_notation_lang_str
2426       }{
2427         \l__stex_notation_symbol_str
2428       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2429     }
2430   }
2431 }
2432 \exp_args:Nx
2433 % \stex_do_up_to_module:n {
2434   \seq_put_right:cx {
2435     l_stex_symdecl_ \l__stex_notation_symbol_str
2436     _notations
2437   } {
2438     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2439   }
2440 % }
2441
2442 \stex_debug:nn{symbols}{
2443   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

2444 ~for~\l__stex_notation_symbol_str^^J
2445 Operator~precedence:~\l__stex_notation_opprec_tl^^J
2446 Argument~precedences:~
2447   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2448 Notation: \cs_meaning:c {
2449   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2450   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2451   _cs
2452 }
2453 }
2454
2455 %\prop_set_eq:cN {
2456 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2457 %   \c_hash_str \l__stex_notation_lang_str _prop
2458 %} \l_tmpb_prop
2459
2460 \exp_args:Ne
2461 \stex_add_to_current_module:n {
2462   \seq_put_right:cn {
2463     l_stex_symdecl_ \l__stex_notation_symbol_str
2464     _notations
2465   } {
2466     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2467   }
2468   %\prop_set_from_keyval:cn {
2469   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2470   %   \c_hash_str \l__stex_notation_lang_str _prop
2471   %} {
2472   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2473   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2474   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2475   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2476   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2477   %}
2478 }
2479
2480 \stex_if_smsmode:F {
2481
2482   % HTML annotations
2483   \stex_if_do_html:T {
2484     \stex_annotate_invisible:nnn { notation }
2485     { \l__stex_notation_symbol_str } {
2486       \stex_annotate_invisible:nnn { notationfragment }
2487       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2488       \stex_annotate_invisible:nnn { precedence }
2489       { \l__stex_notation_prec_str }{}
2490
2491       \int_zero:N \l_tmpa_int
2492       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2493       \tl_clear:N \l_tmpa_tl
2494       \int_step_inline:nn { \l__stex_notation_arity_str }{
2495         \int_incr:N \l_tmpa_int
2496         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2497         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_

```



```

2498 \str_if_eq:VnTF \l_tmpb_str a {
2499 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2500 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2501 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2502 } }
2503 }{
2504 \str_if_eq:VnTF \l_tmpb_str B {
2505 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2506 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2507 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2508 } }
2509 }{
2510 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2511 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2512 } }
2513 }
2514 }
2515 }
2516 \stex_annotate_invisible:nnn { notationcomp }{}{
2517 \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2518 $ \exp_args:Nno \use:nn { \use:c {
2519 stex_notation_ \l_stex_current_symbol_str
2520 \c_hash_str \l__stex_notation_variant_str
2521 \c_hash_str \l__stex_notation_lang_str _cs
2522 } } { \l_tmpa_tl } $
2523 }
2524 }
2525 }
2526 }
2527 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2528 \keys_define:nn { stex / setnotation } {
2529 lang .tl_set_x:N = \l__stex_notation_lang_str ,
2530 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2531 unknown .code:n = \str_set:Nx
2532 \l__stex_notation_variant_str \l_keys_key_str
2533 }
2534
2535 \cs_new_protected:Nn \stex_setnotation_args:n {
2536 \str_clear:N \l__stex_notation_lang_str
2537 \str_clear:N \l__stex_notation_variant_str
2538 \keys_set:nn { stex / setnotation } { #1 }
2539 }
2540
2541 \cs_new_protected:Nn \stex_setnotation:n {
2542 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2543 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2544 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2545 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2546 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2547 { \c_hash_str }

```

```

2548 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2549 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2550 \exp_args:Nx \stex_add_to_current_module:n {
2551 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2552 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2553 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2554 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2555 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2556 { \c_hash_str }
2557 }
2558 \stex_debug:nn {notations}{
2559 Setting~default~notation~
2560 {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2561 #1 \\
2562 \expandafter\meaning\csname
2563 l_stex_symdecl_#1 _notations\endcsname
2564 }
2565 }{
2566 % todo throw error
2567 }
2568 }
2569
2570 \NewDocumentCommand \setnotation {m m} {
2571 \stex_get_symbol:n { #1 }
2572 \_stex_setnotation_args:n { #2 }
2573 \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2574 \stex_smsmode_do:
2575 }
2576
2577 \cs_new_protected:Nn \stex_copy_notations:nn {
2578 \stex_debug:nn {notations}{
2579 Copying~notations~from~#2~to~#1\\
2580 \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2581 }
2582 \tl_clear:N \l_tmpa_tl
2583 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2584 \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2585 }
2586 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2587 \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2588 \edef \l_tmpa_tl {
2589 \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2590 \exp_after:wN\exp_after:wN\exp_after:wN {
2591 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2592 }
2593 }
2594 \exp_args:Nx
2595 \stex_do_up_to_module:n {
2596 \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2597 \cs_generate_from_arg_count:cNnn {
2598 stex_notation_ #1 \c_hash_str ##1 _cs
2599 } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2600 \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2601 }

```

```

2602     }
2603   }
2604 }
2605
2606 \NewDocumentCommand \copynotation {m m} {
2607   \stex_get_symbol:n { #1 }
2608   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2609   \stex_get_symbol:n { #2 }
2610   \exp_args:Noo
2611   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2612   \exp_args:Nx \stex_add_import_to_current_module:n{
2613     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2614   }
2615   \stex_smsmode_do:
2616 }
2617

```

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

\symdef

```

2618 \keys_define:nn { stex / symdef } {
2619   name .str_set_x:N = \l_stex_symdecl_name_str ,
2620   local .bool_set:N = \l_stex_symdecl_local_bool ,
2621   args .str_set_x:N = \l_stex_symdecl_args_str ,
2622   type .tl_set:N = \l_stex_symdecl_type_tl ,
2623   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2624   op .tl_set:N = \l__stex_notation_op_tl ,
2625   lang .str_set_x:N = \l__stex_notation_lang_str ,
2626   variant .str_set_x:N = \l__stex_notation_variant_str ,
2627   prec .str_set_x:N = \l__stex_notation_prec_str ,
2628   assoc .choices:nn =
2629     {bin,binl,binr,pre,conj,pwconj}
2630     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2631   unknown .code:n = \str_set:Nx
2632     \l__stex_notation_variant_str \l_keys_key_str
2633 }
2634
2635 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2636   \str_clear:N \l_stex_symdecl_name_str
2637   \str_clear:N \l_stex_symdecl_args_str
2638   \str_clear:N \l_stex_symdecl_assoctype_str
2639   \bool_set_false:N \l_stex_symdecl_local_bool
2640   \tl_clear:N \l_stex_symdecl_type_tl
2641   \tl_clear:N \l_stex_symdecl_definiens_tl
2642   \str_clear:N \l__stex_notation_lang_str
2643   \str_clear:N \l__stex_notation_variant_str
2644   \str_clear:N \l__stex_notation_prec_str
2645   \tl_clear:N \l__stex_notation_op_tl
2646
2647   \keys_set:nn { stex / symdef } { #1 }
2648 }
2649
2650 \NewDocumentCommand \symdef { O{} m } {
2651   \__stex_notation_symdef_args:n { #1 }

```

```

2652 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2653 \stex_symdecl_do:n { #2 }
2654 \tl_set:Nn \l__stex_notation_after_do_tl {
2655   \__stex_notation_final:
2656   \stex_smsmode_do:
2657 }
2658 \exp_args:Nx \stex_notation_do:nn {
2659   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2660 }
2661 }
2662 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

30.3 Variables

```

2663 <@@=stex_variables>
2664
2665 \keys_define:nn { stex / vardef } {
2666   name .str_set_x:N = \l__stex_variables_name_str ,
2667   args .str_set_x:N = \l__stex_variables_args_str ,
2668   type .tl_set:N    = \l__stex_variables_type_tl ,
2669   def  .tl_set:N    = \l__stex_variables_def_tl ,
2670   op   .tl_set:N    = \l__stex_variables_op_tl ,
2671   prec .str_set_x:N = \l__stex_variables_prec_str ,
2672   assoc .choices:nn =
2673     {bin,binl,binr,pre,conj,pwconj}
2674     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2675   bind .choices:nn =
2676     {forall,exists}
2677     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2678 }
2679
2680 \cs_new_protected:Nn \__stex_variables_args:n {
2681   \str_clear:N \l__stex_variables_name_str
2682   \str_clear:N \l__stex_variables_args_str
2683   \str_clear:N \l__stex_variables_prec_str
2684   \str_clear:N \l__stex_variables_assoctype_str
2685   \str_clear:N \l__stex_variables_bind_str
2686   \tl_clear:N \l__stex_variables_type_tl
2687   \tl_clear:N \l__stex_variables_def_tl
2688   \tl_clear:N \l__stex_variables_op_tl
2689
2690   \keys_set:nn { stex / vardef } { #1 }
2691 }
2692
2693 \NewDocumentCommand \vardecl {0{} m m} {
2694   \__stex_variables_args:n {#1}
2695   \str_if_empty:NT \l__stex_variables_name_str {
2696     \str_set:Nx \l__stex_variables_name_str { #2 }
2697   }
2698   \prop_clear:N \l_tmpa_prop
2699   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2700

```

```

2701 \int_zero:N \l_tmpb_int
2702 \bool_set_true:N \l_tmpa_bool
2703 \str_map_inline:Nn \l__stex_variables_args_str {
2704   \token_case_meaning:NnF ##1 {
2705     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2706     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2707     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2708     {\tl_to_str:n a} {
2709       \bool_set_false:N \l_tmpa_bool
2710       \int_incr:N \l_tmpb_int
2711     }
2712     {\tl_to_str:n B} {
2713       \bool_set_false:N \l_tmpa_bool
2714       \int_incr:N \l_tmpb_int
2715     }
2716   }{
2717     \msg_error:nxxx{stex}{error/wrongargs}{
2718       variable~\l__stex_variables_name_str
2719     }{##1}
2720   }
2721 }
2722 \bool_if:NTF \l_tmpa_bool {
2723   % possibly numeric
2724   \str_if_empty:NTF \l__stex_variables_args_str {
2725     \prop_put:Nnn \l_tmpa_prop { args } {}
2726     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2727   }{
2728     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2729     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2730     \str_clear:N \l_tmpa_str
2731     \int_step_inline:nn \l_tmpa_int {
2732       \str_put_right:Nn \l_tmpa_str i
2733     }
2734     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2735   }
2736 } {
2737   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2738   \prop_put:Nnx \l_tmpa_prop { arity }
2739     { \str_count:N \l__stex_variables_args_str }
2740 }
2741 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2742 \tl_set:cn { #2 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2743
2744
2745
2746
2747
2748
2749 \prop_set_eq:cn { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2750 }
2751
2752
2753
2754

```

2755 `</package>`

Chapter 31

STEX -Terms Implementation

```
2756 <*package>
2757
2758 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2759
2760 <@@=stex_terms>
2761
2762 Warnings and error messages
2763 \msg_new:nnn{stex}{error/nonotation}{
2764   Symbol~#1~invoked,~but~has~no~notation~#2!
2765 }
2766 \msg_new:nnn{stex}{error/notationarg}{
2767   Error~in~parsing~notation~#1
2768 }
2769 \msg_new:nnn{stex}{error/noop}{
2770   Symbol~#1~has~no~operator~notation~for~notation~#2
2771 }
```

31.1 Symbol Invocations

Arguments:

```
2771 \keys_define:nn { stex / terms } {
2772   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2773   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2774   unknown .code:n = \str_set:Nx
2775     \l__stex_terms_variant_str \l_keys_key_str
2776 }
2777
2778 \cs_new_protected:Nn \__stex_terms_args:n {
2779   \str_clear:N \l__stex_terms_lang_str
2780   \str_clear:N \l__stex_terms_variant_str
2781   \str_clear:N \l__stex_terms_prec_str
2782   \tl_clear:N \l__stex_terms_op_tl
2783
2784   \keys_set:nn { stex / terms } { #1 }
```

2785 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2786 \cs_new_protected:Nn \stex_invoke_symbol:n {
2787   \str_if_eq:eeF {
2788     \prop_item:cn {
2789       l_stex_symdecl_#1_prop
2790     }{ deprecate }
2791   }{}{
2792     \msg_warning:nxxx{stex}{warning/deprecated}{
2793       Symbol~#1
2794     }{
2795       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2796     }
2797   }
2798   \if_mode_math:
2799     \exp_after:wN \__stex_terms_invoke_math:n
2800   \else:
2801     \exp_after:wN \__stex_terms_invoke_text:n
2802   \fi: { #1 }
2803 }
```

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

`__stex_terms_invoke_math:n`

```

2804 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2805   \peek_charcode_remove:NTF ! {
2806     \peek_charcode:NTF [ {
2807       \__stex_terms_invoke_op:nw { #1 }
2808     }{
2809       \peek_charcode_remove:NTF ! {
2810         \peek_charcode:NTF [ {
2811           \__stex_terms_invoke_op_custom:nw
2812         }{
2813           % TODO throw error
2814         }
2815       }{
2816         \__stex_terms_invoke_op:nw { #1 } []
2817       }
2818     }
2819   }{
2820     \peek_charcode_remove:NTF * {
2821       \__stex_terms_invoke_text:n { #1 }
2822     }{
2823       \peek_charcode:NTF [ {
2824         \__stex_terms_invoke_math:nw { #1 }
2825       }{
2826         \__stex_terms_invoke_math:nw { #1 } []
2827       }
2828     }
2829   }
2830 }
```

(End definition for `__stex_terms_invoke_math:n`.)

_stex_terms_invoke_op_custom:nw

```

2831 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2832   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2833     \stex_highlight_term:nn{#1}{#2}
2834   }
2835 }

```

(End definition for _stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2836 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2837   \_stex_terms_args:n { #2 }
2838   \cs_if_exist:cTF {
2839     stex_op_notation_ #1 \c_hash_str
2840     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2841   }{
2842     \csname stex_op_notation_ #1 \c_hash_str
2843       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2844     \endcsname
2845   }{
2846     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2847   }
2848 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2849 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2850   \_stex_terms_args:n { #2 }
2851   \seq_if_empty:cTF {
2852     l_stex_symdecl_ #1 _notations
2853   } {
2854     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2855   } {
2856     \seq_if_in:cxTF {
2857       l_stex_symdecl_ #1 _notations
2858     }
2859     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2860       \str_set:Nn \l_stex_current_symbol_str { #1 }
2861       \stex_debug:nn{terms}{Using~
2862         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2863         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2864         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2865         _cs\endcsname
2866       }
2867       \use:c{
2868         stex_notation_ #1 \c_hash_str
2869         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2870         _cs
2871       }
2872     }{
2873       \str_if_empty:NTF \l__stex_terms_variant_str {
2874         \str_if_empty:NTF \l__stex_terms_lang_str {
2875           \seq_get_left:cN {

```

```

2876         l_stex_symdecl_ #1 _notations
2877     } \l_tmpa_str
2878     \str_set:Nn \l_stex_current_symbol_str { #1 }
2879     \stex_debug:nn{terms}{Using~
2880         #1\c_hash_str\l_tmpa_str \\
2881         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2882         \l_tmpa_str
2883         _cs\endcsname
2884     }
2885     \use:c{
2886         stex_notation_ #1 \c_hash_str \l_tmpa_str
2887         _cs
2888     }
2889   }{
2890     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2891       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2892     }
2893   }
2894   }{
2895     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2896       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2897     }
2898   }
2899 }
2900 }
2901 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2902 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2903   \peek_charcode_remove:NTF ! {
2904     \stex_term_custom:nn { #1 } { }
2905   }{
2906     \prop_set_eq:Nc \l_tmpa_prop {
2907       l_stex_symdecl_ #1 _prop
2908     }
2909     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2910     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2911   }
2912 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2913 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2914 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2915 \int_new:N \l__stex_terms_downprec
2916 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

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

Bracketing:

`\l__stex_terms_left_bracket_str`
`\l__stex_terms_right_bracket_str`

```
2917 \tl_set:Nn \l__stex_terms_left_bracket_str (
2918 \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

```
2919 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2920   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2921     \bool_set_false:N \l__stex_terms_brackets_done_bool
2922     #2
2923   } {
2924     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2925       \bool_if:NTF \l__stex_inarray_bool { #2 } {
2926         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2927         \dobrackets { #2 }
2928       }
2929     }{ #2 }
2930   }
2931 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2932 \bool_new:N \l__stex_terms_brackets_done_bool
2933 %\RequirePackage{scalerel}
2934 \cs_new_protected:Npn \dobrackets #1 {
2935   %\ThisStyle{\if D\m@switch
2936   %   \exp_args:Nnx \use:nn
2937   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2938   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2939   %   \else
2940   %   \exp_args:Nnx \use:nn
2941   %   {
2942     \bool_set_true:N \l__stex_terms_brackets_done_bool
2943     \int_set:Nn \l__stex_terms_downprec \infprec
2944     \l__stex_terms_left_bracket_str
2945     #1
2946   }
2947   {
2948     \bool_set_false:N \l__stex_terms_brackets_done_bool
2949     \l__stex_terms_right_bracket_str
2950     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2951   }
2952   %\fi}
2953 }
```

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

\withbrackets

```
2954 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2955   \exp_args:Nnx \use:nn
2956   {
2957     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2958     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2959     #3
2960   }
2961   {
2962     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2963       {\l__stex_terms_left_bracket_str}
2964     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2965       {\l__stex_terms_right_bracket_str}
2966   }
2967 }
```

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

\STEXinvisible

```
2968 \cs_new_protected:Npn \STEXinvisible #1 {
2969   \stex_annotate_invisible:n { #1 }
2970 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
2971 \cs_new_protected:Nn \_stex_term_oms:nnn {
2972   \stex_annotate:nnn{ OMID }{ #2 }{
2973     \stex_highlight_term:nn { #1 } { #3 }
2974   }
2975 }
2976
2977 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2978   \__stex_terms_maybe_brackets:nn { #3 }{
2979     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2980   }
2981 }
```

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

_stex_term_math_oma:nnnn

```
2982 \cs_new_protected:Nn \_stex_term_oma:nnn {
2983   \stex_annotate:nnn{ OMA }{ #2 }{
2984     \stex_highlight_term:nn { #1 } { #3 }
2985   }
2986 }
2987
2988 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2989   \__stex_terms_maybe_brackets:nn { #3 }{
2990     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2991   }
2992 }
```

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

`_stex_term_math_omb:nnnn`

```
2993 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2994   \stex_annotate:nnn{ OMBIND }{ #2 }{
2995     \stex_highlight_term:nn { #1 } { #3 }
2996   }
2997 }
2998
2999 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3000   \__stex_terms_maybe_brackets:nn { #3 }{
3001     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3002   }
3003 }
```

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

`_stex_term_math_arg:nnn`

```
3004 \cs_new_protected:Nn \_stex_term_arg:nn {
3005   \stex_unhighlight_term:n {
3006     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3007   }
3008 }
3009 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3010   \exp_args:Nnx \use:nn
3011   { \int_set:Nn \l__stex_terms_downprec { #2 }
3012     \_stex_term_arg:nn { #1 }{ #3 }
3013   }
3014   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3015 }
```

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

`_stex_term_math_assoc_arg:nnnn`

```
3016 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3017   % TODO sequences
3018   \clist_set:Nn \l_tmpa_clist{ #3 }
3019   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3020     \tl_set:Nn \l_tmpa_tl { #3 }
3021   }{
3022     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3023     \clist_reverse:N \l_tmpa_clist
3024     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3025
3026     \clist_map_inline:Nn \l_tmpa_clist {
3027       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3028         \exp_args:Nno
3029         \l_tmpa_cs { ##1 } \l_tmpa_tl
3030       }
3031     }
3032   }
3033   \exp_args:Nnno
3034   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3035 }
```

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

`\stex_term_custom:nn`

```

3036 \cs_new_protected:Nn \stex_term_custom:nn {
3037   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3038   \str_set:Nn \l_tmpa_str { #2 }
3039   \tl_clear:N \l_tmpa_tl
3040   \int_zero:N \l_tmpa_int
3041   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3042   \__stex_terms_custom_loop:
3043 }

```

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

`__stex_terms_custom_loop:`

```

3044 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3045   \bool_set_false:N \l_tmpa_bool
3046   \bool_while_do:nn {
3047     \str_if_eq_p:ee X {
3048       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3049     }
3050   }{
3051     \int_incr:N \l_tmpa_int
3052   }
3053
3054   \peek_charcode:NTF [ {
3055     % notation/text component
3056     \__stex_terms_custom_component:w
3057   } {
3058     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3059       % all arguments read => finish
3060       \__stex_terms_custom_final:
3061     } {
3062       % arguments missing
3063       \peek_charcode_remove:NTF * {
3064         % invisible, specific argument position or both
3065         \peek_charcode:NTF [ {
3066           % visible specific argument position
3067           \__stex_terms_custom_arg:wn
3068         } {
3069           % invisible
3070           \peek_charcode_remove:NTF * {
3071             % invisible specific argument position
3072             \__stex_terms_custom_arg_inv:wn
3073           } {
3074             % invisible next argument
3075             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3076           }
3077         }
3078       } {
3079         % next normal argument
3080         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3081       }
3082     }
3083   }
3084 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

3085 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3086   \bool_set_true:N \l_tmpa_bool
3087   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3088 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

3089 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3090   \str_set:Nx \l_tmpb_str {
3091     \str_item:Nn \l_tmpa_str { #1 }
3092   }
3093   \str_case:VnTF \l_tmpb_str {
3094     { X } {
3095       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3096     }
3097     { i } { \_stex_terms_custom_set_X:n { #1 } }
3098     { b } { \_stex_terms_custom_set_X:n { #1 } }
3099     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3100     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3101   }{}{
3102     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3103   }
3104
3105   \bool_if:nTF \l_tmpa_bool {
3106     \tl_put_right:Nx \l_tmpa_tl {
3107       \stex_annotate_invisible:n {
3108         \_stex_term_arg:nn { \int_eval:n { #1 } }
3109         \exp_not:n { { #2 } }
3110       }
3111     }
3112   } {
3113     \tl_put_right:Nx \l_tmpa_tl {
3114       \_stex_term_arg:nn { \int_eval:n { #1 } }
3115       \exp_not:n { { #2 } }
3116     }
3117   }
3118
3119   \_stex_terms_custom_loop:
3120 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

3121 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3122   \str_set:Nx \l_tmpa_str {
3123     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3124     X
3125     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3126   }
3127 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

3128 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3129   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3130   \_stex_terms_custom_loop:
3131 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

3132 \cs_new_protected:Nn \_stex_terms_custom_final: {
3133   \int_compare:nNnTF \l_tmpb_int = 0 {
3134     \exp_args:Nnno \_stex_term_oms:nnn
3135   }{
3136     \str_if_in:NnTF \l_tmpa_str {b} {
3137       \exp_args:Nnno \_stex_term_ombind:nnn
3138     } {
3139       \exp_args:Nnno \_stex_term_oma:nnn
3140     }
3141   }
3142   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3143 }

```

(End definition for _stex_terms_custom_final:.)

\symref
\symname

```

3144 \NewDocumentCommand \symref { m m }{
3145   \let\compemph_uri_prev:\compemph@uri
3146   \let\compemph@uri\symrefemph@uri
3147   \STEXsymbol{#1}!{#2}
3148   \let\compemph@uri\compemph_uri_prev:
3149 }
3150
3151 \keys_define:nn { stex / symname } {
3152   post .str_set_x:N = \l_stex_symname_post_str
3153 }
3154
3155 \cs_new_protected:Nn \stex_symname_args:n {
3156   \str_clear:N \l_stex_symname_post_str
3157   \keys_set:nn { stex / symname } { #1 }
3158 }
3159
3160 \NewDocumentCommand \symname { 0{} m }{
3161   \stex_symname_args:n { #1 }
3162   \stex_get_symbol:n { #2 }
3163   \str_set:Nx \l_tmpa_str {
3164     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3165   }
3166   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3167
3168   \let\compemph_uri_prev:\compemph@uri
3169   \let\compemph@uri\symrefemph@uri
3170   \exp_args:NNx \use:nn

```



```

3171 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3172 \l_tmpa_str \l_stex_symname_post_str
3173 ] }
3174 \let\compemph@uri\compemph_uri_prev:
3175 }

```

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

31.3 Notation Components

```

3176 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3177
3178 \str_new:N \l_stex_current_symbol_str
3179 \cs_new_protected:Nn \stex_highlight_term:nn {
3180 \exp_args:Nnx
3181 \use:nn {
3182 \str_set:Nx \l_stex_current_symbol_str { #1 }
3183 #2
3184 } {
3185 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3186 { \l_stex_current_symbol_str }
3187 }
3188 }
3189
3190 \cs_new_protected:Nn \stex_unhighlight_term:n {
3191 % \latexml_if:TF {
3192 % #1
3193 % } {
3194 % \rustex_if:TF {
3195 % #1
3196 % } {
3197 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3198 % }
3199 % }
3200 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3201 \cs_new_protected:Npn \comp #1 {
3202 \str_if_empty:NF \l_stex_current_symbol_str {
3203 \rustex_if:TF {
3204 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3205 }{
3206 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3207 }
3208 }
3209 }
3210
3211 \cs_new_protected:Npn \compemph@uri #1 #2 {
3212 \compemph{ #1 }
3213 }

```

```

3214
3215
3216 \cs_new_protected:Npn \compemph #1 {
3217     #1
3218 }
3219
3220 \cs_new_protected:Npn \defemph@uri #1 #2 {
3221     \defemph{#1}
3222 }
3223
3224 \cs_new_protected:Npn \defemph #1 {
3225     \textbf{#1}
3226 }
3227
3228 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3229     \symrefemph{#1}
3230 }
3231
3232 \cs_new_protected:Npn \symrefemph #1 {
3233     \textbf{#1}
3234 }

```

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

`\ellipses`

```

3235 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3236 \bool_new:N \l_stex_inarray_bool
3237 \bool_set_false:N \l_stex_inarray_bool
3238 \NewDocumentCommand \parray { m m } {
3239     \begingroup
3240     \bool_set_true:N \l_stex_inarray_bool
3241     \begin{array}{#1}
3242         #2
3243     \end{array}
3244     \endgroup
3245 }
3246
3247 \NewDocumentCommand \prmatrix { m } {
3248     \begingroup
3249     \bool_set_true:N \l_stex_inarray_bool
3250     \begin{matrix}
3251         #1
3252     \end{matrix}
3253     \endgroup
3254 }
3255
3256 \def \maybepline {
3257     \bool_if:NT \l_stex_inarray_bool {\hline}
3258 }
3259
3260 \def \parrayline #1 #2 {

```

```

3261   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3262 }
3263
3264 \def \pmrow #1 { \parrayline{ }{ #1 } }
3265
3266 \def \parraylineh #1 #2 {
3267   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3268 }
3269
3270 \def \parraycell #1 {
3271   #1 \bool_if:NT \l_stex_inarray_bool {&}
3272 }

```

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

```

3273 \endpackage

```

Chapter 32

STEX -Structural Features Implementation

```
3274 <*package>
3275
3276 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3277
3278 <@@=stex_features>
3279
3280 Warnings and error messages
3281 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3282   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3283 }
3284 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3285   Symbol~#1~not~assigned~in~interpretmodule~#2
3286 }
```

32.1 Imports with modification

```
3286 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3287   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3288     \__stex_features_get_symbol_from_cs:n { #1 }
3289   }{
3290     % argument is a string
3291     % is it a command name?
3292     \cs_if_exist:cTF { #1 }{
3293       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3294       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3295       \str_if_empty:NNTF \l_tmpa_str {
3296         \exp_args:Nx \cs_if_eq:NNTF {
3297           \tl_head:N \l_tmpa_tl
3298         } \stex_invoke_symbol:n {
3299           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3300         }{
3301           \__stex_features_get_symbol_from_string:n { #1 }
3302         }
3303       }
3304     }
3305   }
```

```

3302     }
3303   } {
3304     \__stex_features_get_symbol_from_string:n { #1 }
3305   }
3306   }{
3307     % argument is not a command name
3308     \__stex_features_get_symbol_from_string:n { #1 }
3309     % \l_stex_all_symbols_seq
3310   }
3311 }
3312 }
3313
3314 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3315   \str_set:Nn \l_tmpa_str { #1 }
3316   \bool_set_false:N \l_tmpa_bool
3317   \bool_if:NF \l_tmpa_bool {
3318     \tl_set:Nn \l_tmpa_tl {
3319       \msg_set:nnn{stex}{error/unknownsymbol}{
3320         No~symbol~#1~found!
3321       }
3322       \msg_error:nn{stex}{error/unknownsymbol}
3323     }
3324     \str_set:Nn \l_tmpa_str { #1 }
3325     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3326     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3327       \str_set:Nn \l_tmpb_str { ##1 }
3328       \str_if_eq:eeT { \l_tmpa_str } {
3329         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3330       } {
3331         \seq_map_break:n {
3332           \tl_set:Nn \l_tmpa_tl {
3333             \str_set:Nn \l_stex_get_symbol_uri_str {
3334               ##1
3335             }
3336             \__stex_features_get_symbol_check:
3337           }
3338         }
3339       }
3340     }
3341     \l_tmpa_tl
3342   }
3343 }
3344
3345 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3346   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3347   { \tl_tail:N \l_tmpa_tl }
3348   \tl_if_single:NTF \l_tmpa_tl {
3349     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3350       \exp_after:wN \str_set:Nn \exp_after:wN
3351       \l_stex_get_symbol_uri_str \l_tmpa_tl
3352       \__stex_features_get_symbol_check:
3353     }{
3354       % TODO
3355       % tail is not a single group

```

```

3356     }
3357   }{
3358     % TODO
3359     % tail is not a single group
3360   }
3361 }
3362
3363 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3364   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3365   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3366     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3367     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3368     \seq_if_in:Nof \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3369       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3370         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3371       }
3372     }
3373   }{
3374     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3375       \l_stex_current_copymodule_name_str~(inexplicably)
3376     }
3377   }
3378 }
3379
3380 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3381   \stex_import_module_uri:nn { #1 } { #2 }
3382   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3383   \stex_import_require_module:nnnn
3384     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3385     { \l_stex_import_path_str } { \l_stex_import_name_str }
3386   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3387   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3388   \seq_clear:N \l__stex_features_copymodule_fields_seq
3389   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3390     \seq_map_inline:cn {c_stex_module_###1_constants}{
3391       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3392         ###1 ? ####1
3393       }
3394     }
3395   }
3396   \seq_clear:N \l_tmpa_seq
3397   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3398     name      = \l_stex_current_copymodule_name_str ,
3399     module    = \l_stex_current_module_str ,
3400     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3401     includes  = \l_tmpa_seq ,
3402     fields    = \l_tmpa_seq
3403   }
3404   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3405     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3406   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3407     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3408   \stex_if_smsmode:F {
3409     \begin{stex_annotate_env} {#4} {

```

```

3410     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3411   }
3412   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3413 }
3414 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3415 \bool_set_false:N \stex_html_do_output_bool
3416 }
3417 \cs_new_protected:Nn \stex_copymodule_end:n {
3418   \def \l_tmpa_cs ##1 ##2 {#1}
3419   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3420   \tl_clear:N \l_tmpa_tl
3421   \tl_clear:N \l_tmpb_tl
3422   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3423   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3424     \seq_map_inline:cn {c_stex_module_##1_constants}{
3425       \tl_clear:N \l_tmpc_tl
3426       \l_tmpa_cs{##1}{####1}
3427       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3428         \tl_put_right:Nx \l_tmpa_tl {
3429           \prop_set_from_keyval:cn {
3430             l_stex_symdecl_\l_stex_current_module_str ? \use:c{\l__stex_features_copymodule_#
3431           }{
3432             \exp_after:wN \prop_to_keyval:N \csname
3433               l_stex_symdecl_\l_stex_current_module_str ? \use:c{\l__stex_features_copymodule_#
3434             \endcsname
3435           }
3436           \seq_clear:c {
3437             l_stex_symdecl_
3438             \l_stex_current_module_str ? \use:c{\l__stex_features_copymodule_##1?####1_name_s
3439             _notations
3440           }
3441         }
3442         \tl_put_right:Nx \l_tmpc_tl {
3443           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{\l__stex_features_cop
3444           \stex_annotate_invisible:nnn{alias}{\use:c{\l__stex_features_copymodule_##1?####1_n
3445         }
3446         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{\l__stex_features_
3447         \str_if_exist:cT {\l__stex_features_copymodule_##1?####1_macroname_str} {
3448           \tl_put_right:Nx \l_tmpc_tl {
3449             \stex_annotate_invisible:nnn{macroname}{\use:c{\l__stex_features_copymodule_##1?#
3450           }
3451           \tl_put_right:Nx \l_tmpa_tl {
3452             \tl_set:cx {\use:c{\l__stex_features_copymodule_##1?####1_macroname_str}}{
3453             \stex_invoke_symbol:n {
3454               \l_stex_current_module_str ? \use:c{\l__stex_features_copymodule_##1?####1_na
3455             }
3456           }
3457         }
3458       }
3459     }{
3460       \tl_put_right:Nx \l_tmpc_tl {
3461         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3462       }
3463       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3464 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3465 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3466 \tl_put_right:Nx \l_tmpa_tl {
3467   \prop_set_from_keyval:cn {
3468     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3469   }{
3470     \prop_to_keyval:N \l_tmpa_prop
3471   }
3472   \seq_clear:c {
3473     l_stex_symdecl_
3474     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3475     _notations
3476   }
3477 }
3478 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3479 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3480   \tl_put_right:Nx \l_tmpc_tl {
3481     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3482   }
3483   \tl_put_right:Nx \l_tmpa_tl {
3484     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3485       \stex_invoke_symbol:n {
3486         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3487       }
3488     }
3489   }
3490 }
3491 }
3492 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3493   \tl_put_right:Nx \l_tmpc_tl {
3494     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3495   }
3496 }
3497 \tl_put_right:Nx \l_tmpb_tl {
3498   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3499 }
3500 }
3501 }
3502 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3503 \tl_put_left:Nx \l_tmpa_tl {
3504   \prop_set_from_keyval:cn {
3505     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3506   }{
3507     \prop_to_keyval:N \l_stex_current_copymodule_prop
3508   }
3509 }
3510 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3511 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3512 \exp_args:Nx \stex_do_up_to_module:n {
3513   \exp_args:No \exp_not:n \l_tmpa_tl
3514 }
3515 \l_tmpb_tl
3516 \stex_if_smsmode:F {
3517   \end{stex_annotate_env}

```



```

3518 }
3519 }
3520
3521 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3522   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3523   \stex_deactivate_macro:Nn \symdecl {module~environments}
3524   \stex_deactivate_macro:Nn \symdef {module~environments}
3525   \stex_deactivate_macro:Nn \notation {module~environments}
3526   \stex_reactivate_macro:N \assign
3527   \stex_reactivate_macro:N \renamedec1
3528   \stex_reactivate_macro:N \donotcopy
3529   \stex_smsmode_do:
3530 }{
3531   \stex_copymodule_end:n {}
3532 }
3533
3534 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3535   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3536   \stex_deactivate_macro:Nn \symdecl {module~environments}
3537   \stex_deactivate_macro:Nn \symdef {module~environments}
3538   \stex_deactivate_macro:Nn \notation {module~environments}
3539   \stex_reactivate_macro:N \assign
3540   \stex_reactivate_macro:N \renamedec1
3541   \stex_reactivate_macro:N \donotcopy
3542   \stex_smsmode_do:
3543 }{
3544   \stex_copymodule_end:n {
3545     \tl_if_exist:cF {
3546       l__stex_features_copymodule_##1?##2_def_tl
3547     }{
3548       \msg_error:nxxx{stex}{error/interpretmodule/nodedefiniens}{
3549         ##1?##2
3550       }{\l_stex_current_copymodule_name_str}
3551     }
3552   }
3553 }
3554
3555 \NewDocumentCommand \donotcopy { 0{} m}{
3556   \stex_import_module_uri:nn { #1 } { #2 }
3557   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3558   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3559     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3560     \seq_map_inline:cn {c_stex_module_##1_constants}{
3561       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3562       \bool_lazy_any_p:nT {
3563         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3564         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3565         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3566       }{
3567         % TODO throw error
3568       }
3569     }
3570   }
3571 }

```

```

3572 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3573 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3574 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3575 }
3576
3577 \NewDocumentCommand \assign { m m }{
3578   \stex_get_symbol_in_copymodule:n {#1}
3579   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3580   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3581 }
3582
3583 \keys_define:nn { stex / renamedec1 } {
3584   name .str_set_x:N = \l_stex_renamedec1_name_str
3585 }
3586 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3587   \str_clear:N \l_stex_renamedec1_name_str
3588
3589   \keys_set:nn { stex / renamedec1 } { #1 }
3590 }
3591
3592 \NewDocumentCommand \renamedec1 { 0{} m m }{
3593   \__stex_features_renamedec1_args:n { #1 }
3594   \stex_get_symbol_in_copymodule:n {#2}
3595   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3596   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3597   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3598     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3599       \l_stex_get_symbol_uri_str
3600     } }
3601   } {
3602     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3603     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3604     \prop_set_eq:cc {l_stex_symdecl_
3605       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3606     _prop
3607     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3608     \seq_set_eq:cc {l_stex_symdecl_
3609       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3610     _notations
3611     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3612     \prop_put:cnx {l_stex_symdecl_
3613       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3614     _prop
3615     }{ name }{ \l_stex_renamedec1_name_str }
3616     \prop_put:cnx {l_stex_symdecl_
3617       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3618     _prop
3619     }{ module }{ \l_stex_current_module_str }
3620     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3621       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3622     }
3623     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3624       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3625     } }

```

```

3626 }
3627 }
3628 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3629 % \_stex_notation_args:n { #1 }
3630 % \tl_clear:N \l_stex_symdecl_definiens_tl
3631 % \stex_get_symbol_in_copymodule:n { #2 }
3632 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3633 % % todo
3634 %}
3635 \stex_deactivate_macro:Nn \assign {copymodules}
3636 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3637 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3638
3639
3640 \seq_new:N \l_stex_implicit_morphisms_seq
3641 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3642 \stex_import_module_uri:nn { #1 } { #2 }
3643 \stex_debug:nn{implicits}{
3644 Implicit~morphism:~
3645 \l_stex_module_ns_str ? \l__stex_features_name_str
3646 }
3647 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3648 \l_stex_module_ns_str ? \l__stex_features_name_str
3649 }{
3650 \msg_error:nnn{stex}{error/conflictingmodules}{
3651 \l_stex_module_ns_str ? \l__stex_features_name_str
3652 }
3653 }
3654
3655 % TODO
3656
3657
3658
3659 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3660 \l_stex_module_ns_str ? \l__stex_features_name_str
3661 }
3662 }
3663

```

32.2 The feature environment

structural@feature

```

3664
3665 \NewDocumentEnvironment{structural@feature}{ m m m }{
3666 \stex_if_in_module:F {
3667 \msg_set:nnn{stex}{error/nomodule}{
3668 Structural~Feature~has~to~occur~in~a~module:\\
3669 Feature~#2~of~type~#1\\
3670 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3671 }
3672 \msg_error:nn{stex}{error/nomodule}
3673 }
3674

```

```

3675 \str_set:Nx \l_stex_module_name_str {
3676   \prop_item:Nn \l_stex_current_module_prop
3677     { name } / #2 - feature
3678 }
3679
3680 \str_set:Nx \l_stex_module_ns_str {
3681   \prop_item:Nn \l_stex_current_module_prop
3682     { ns }
3683 }
3684
3685
3686 \str_clear:N \l_tmpa_str
3687 \seq_clear:N \l_tmpa_seq
3688 \tl_clear:N \l_tmpa_tl
3689 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3690   origname = #2,
3691   name     = \l_stex_module_name_str ,
3692   ns       = \l_stex_module_ns_str ,
3693   imports  = \exp_not:o { \l_tmpa_seq } ,
3694   constants = \exp_not:o { \l_tmpa_seq } ,
3695   content  = \exp_not:o { \l_tmpa_tl } ,
3696   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3697   lang     = \l_stex_module_lang_str ,
3698   sig      = \l_tmpa_str ,
3699   meta     = \l_tmpa_str ,
3700   feature  = #1 ,
3701 }
3702
3703 \stex_if_smsmode:F {
3704   \begin{stex_annotate_env}{ feature:#1 }{}
3705   \stex_annotate_invisible:nnn{header}{}{ #3 }
3706 }
3707 }{
3708   \str_set:Nx \l_tmpa_str {
3709     c_stex_feature_
3710     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3711     \prop_item:Nn \l_stex_current_module_prop { name }
3712     _prop
3713   }
3714   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3715   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3716   \stex_if_smsmode:F {
3717     \end{stex_annotate_env}
3718   }
3719 }
3720

```

32.3 Features

structure

```

3721
3722 \prop_new:N \l_stex_all_structures_prop
3723

```

```

3724 \keys_define:nn { stex / features / structure } {
3725   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3726 }
3727
3728 \cs_new_protected:Nn \__stex_features_structure_args:n {
3729   \str_clear:N \l__stex_features_structure_name_str
3730   \keys_set:nn { stex / features / structure } { #1 }
3731 }
3732
3733 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3734 %   \__stex_features_structure_args:n { ##1 }
3735 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3736 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3737 %   }
3738 % } {
3739 %
3740 %}
3741
3742 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3743   \__stex_features_structure_args:n { #1 }
3744   \str_if_empty:NT \l__stex_features_structure_name_str {
3745     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3746   }
3747   \exp_args:Nnnx
3748   \begin{structural@feature}{ structure }
3749     { \l__stex_features_structure_name_str }{}
3750     \seq_clear:N \l_tmpa_seq
3751     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3752     \stex_smsmode_do:
3753   }{
3754     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3755     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3756     \str_set:Nx \l_tmpa_str {
3757       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3758       \prop_item:Nn \l_stex_current_module_prop { name }
3759     }
3760     \seq_map_inline:Nn \l_tmpa_seq {
3761       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3762     }
3763     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3764     \exp_args:Nnx
3765     \AddToHookNext { env / mathstructure / after }{
3766       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3767         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
3768       }, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3769     \STEXexport {
3770       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3771       { \prop_item:Nn \l_stex_current_module_prop { origname } }
3772       { \l_tmpa_str }
3773       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3774       { #2 } { \l_tmpa_str }
3775     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3776     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3777     %     \l_tmpa_str

```

```

3778 %      }
3779 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3780 %          #2,\l_tmpa_str
3781 %      }
3782 %      \tl_set:cx { #2 } {
3783 %          \stex_invoke_structure:n { \l_tmpa_str }
3784 %      }
3785 %  }
3786
3787 \end{structural@feature}
3788 % \g_stex_last_feature_prop
3789 }

```

\instantiate

```

3790 \seq_new:N \l__stex_features_structure_field_seq
3791 \str_new:N \l__stex_features_structure_field_str
3792 \str_new:N \l__stex_features_structure_def_tl
3793 \prop_new:N \l__stex_features_structure_prop
3794 \NewDocumentCommand \instantiate { m O{} m }{
3795   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3796   \prop_set_eq:Nc \l__stex_features_structure_prop {
3797     c_stex_feature_\l_tmpa_str _prop
3798   }
3799   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3800   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3801     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3802     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3803       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3804       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3805         {!} \l_tmpa_tl
3806       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3807         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3808         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3809         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3810       }{
3811         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3812         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3813         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3814           \l_tmpa_tl
3815         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3816           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3817           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3818         }{
3819           \tl_clear:N \l_tmpb_tl
3820         }
3821       }
3822     }{
3823       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3824       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3825         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3826         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3827         \tl_clear:N \l_tmpa_tl
3828       }{
3829         % TODO throw error

```

```

3830     }
3831   }
3832   % \l_tmpa_str: name
3833   % \l_tmpa_tl: definiens
3834   % \l_tmpb_tl: notation
3835   \tl_if_empty:NT \l__stex_features_structure_field_str {
3836     % TODO throw error
3837   }
3838   \str_clear:N \l_tmpb_str
3839
3840   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3841   \seq_map_inline:Nn \l_tmpa_seq {
3842     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3843     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3844     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3845       \seq_map_break:n {
3846         \str_set:Nn \l_tmpb_str { ####1 }
3847       }
3848     }
3849   }
3850   \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3851   \l_tmpb_str
3852
3853   \tl_if_empty:NTF \l_tmpb_tl {
3854     \tl_if_empty:NF \l_tmpa_tl {
3855       \exp_args:Nx \use:n {
3856         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3857       }
3858     }
3859   }{
3860     \tl_if_empty:NTF \l_tmpa_tl {
3861       \exp_args:Nx \use:n {
3862         \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3863       }
3864     }
3865   }{
3866     \exp_args:Nx \use:n {
3867       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3868       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3869     }
3870   }
3871 }
3872 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3873 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3874 % #3/\l__stex_features_structure_field_str
3875 % \par
3876 % \expandafter\present\csname
3877 %   l_stex_symdecl_
3878 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3879 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3880 %   #3/\l__stex_features_structure_field_str
3881 %   _prop
3882 % \endcsname
3883 }

```

```

3884
3885 \tl_clear:N \l__stex_features_structure_def_tl
3886
3887 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3888 \seq_map_inline:Nn \l_tmpa_seq {
3889   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3890   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3891   \exp_args:Nx \use:n {
3892     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3893
3894     }
3895   }
3896
3897   \prop_if_exist:cF {
3898     l_stex_symdecl_
3899     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3900     \prop_item:Nn \l_stex_current_module_prop {name} ?
3901     #3/\l_tmpa_str
3902     _prop
3903   }{
3904     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3905     \l_tmpb_str
3906     \exp_args:Nx \use:n {
3907       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3908     }
3909   }
3910 }
3911
3912 \symdecl*[type={\STEXsymbol{module-type}}{
3913   \_stex_term_math_oms:nnnn {
3914     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3915     \prop_item:Nn \l__stex_features_structure_prop {name}
3916   }{0}{0}
3917 }]{#3}
3918
3919 % TODO: -> sms file
3920
3921 \tl_set:cx{ #3 }{
3922   \stex_invoke_structure:nnn {
3923     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3924     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3925   } {
3926     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3927     \prop_item:Nn \l__stex_features_structure_prop {name}
3928   }
3929 }
3930 \stex_smsmode_do:
3931 }

```

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

\stex_invoke_structure:nnn

```

3932 % #1: URI of the instance
3933 % #2: URI of the instantiated module

```



```

3934 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3935   \tl_if_empty:nTF{ #3 }{
3936     \prop_set_eq:Nc \l__stex_features_structure_prop {
3937       c_stex_feature_ #2 _prop
3938     }
3939     \tl_clear:N \l_tmpa_tl
3940     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3941     \seq_map_inline:Nn \l_tmpa_seq {
3942       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3943       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3944       \cs_if_exist:cT {
3945         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3946       }{
3947         \tl_if_empty:NF \l_tmpa_tl {
3948           \tl_put_right:Nn \l_tmpa_tl {,}
3949         }
3950         \tl_put_right:Nx \l_tmpa_tl {
3951           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3952         }
3953       }
3954     }
3955     \exp_args:No \mathstrut \l_tmpa_tl
3956   }{
3957     \stex_invoke_symbol:n{#1/#3}
3958   }
3959 }

```

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

```

3960 </package>

```

Chapter 33

STEX -Statements Implementation

```
3961 <*package>
3962
3963 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3964
3965 <@@=stex_statements>
3966
3967 Warnings and error messages
3968
3969 \titleemph
3970
3971 \def\titleemph#1{\textbf{#1}}
3972
3973 (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
3968 \keys_define:nn {stex / definiendum }{
3969   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3970   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3971   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3972 }
3973 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3974   \str_clear:N \l__stex_statements_definiendum_root_str
3975   \tl_clear:N \l__stex_statements_definiendum_post_tl
3976   \str_clear:N \l__stex_statements_definiendum_gfa_str
3977   \keys_set:nn { stex / definiendum }{ #1 }
3978 }
3979 \NewDocumentCommand \definiendum { O{} m m } {
3980   \__stex_statements_definiendum_args:n { #1 }
3981   \stex_get_symbol:n { #2 }
3982   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
3983   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3984     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3985       \tl_set:Nn \l_tmpa_tl { #3 }
3986     }
3987   }
3988 }
```

```

3986   } {
3987     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3988     \tl_set:Nn \l_tmpa_tl {
3989       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3990     }
3991   }
3992 } {
3993   \tl_set:Nn \l_tmpa_tl { #3 }
3994 }
3995
3996 % TODO root
3997 \rustex_if:TF {
3998   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3999 } {
4000   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4001 }
4002 }
4003 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4004
4005 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4006
4007 \NewDocumentCommand \definame { 0{ } m } {
4008   \__stex_statements_definiendum_args:n { #1 }
4009   % TODO: root
4010   \stex_get_symbol:n { #2 }
4011   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4012   \str_set:Nx \l_tmpa_str {
4013     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4014   }
4015   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4016   \rustex_if:TF {
4017     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4018       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4019     }
4020   } {
4021     \defemph@uri {
4022       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4023     } { \l_stex_get_symbol_uri_str }
4024   }
4025 }
4026 \stex_deactivate_macro:Nn \definame {definition~environments}
4027
4028 \NewDocumentCommand \Definame { 0{ } m } {
4029   \__stex_statements_definiendum_args:n { #1 }
4030   \stex_get_symbol:n { #2 }
4031   \str_set:Nx \l_tmpa_str {
4032     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4033   }
4034   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4035   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4036 \rustex_if:TF {
4037   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4038     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4039   }
4040 } {
4041   \defemph@uri {
4042     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4043   } { \l_stex_get_symbol_uri_str }
4044 }
4045 }
4046 \stex_deactivate_macro:Nn \Definame {definition-environments}
4047
4048 \NewDocumentCommand \Symname { 0{ } m }{
4049   \stex_symname_args:n { #1 }
4050   \stex_get_symbol:n { #2 }
4051   \str_set:Nx \l_tmpa_str {
4052     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4053   }
4054   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4055   \let\compemph_uri_prev:\compemph@uri
4056   \let\compemph@uri\symrefemph@uri
4057   \exp_args:NNx \use:nn
4058   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
4059     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4060     \l_stex_symname_post_str
4061   ] }
4062   \let\compemph@uri\compemph_uri_prev:
4063 }

```

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

sdefinition

```

4064
4065 \keys_define:nn {stex / sdefinition }{
4066   type      .str_set_x:N = \sdefinitiontype,
4067   id        .str_set_x:N = \sdefinitionid,
4068   name      .str_set_x:N = \sdefinitionname,
4069   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4070   title     .tl_set:N     = \sdefinitiontitle
4071 }
4072 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4073   \str_clear:N \sdefinitiontype
4074   \str_clear:N \sdefinitionid
4075   \str_clear:N \sdefinitionname
4076   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4077   \tl_clear:N \sdefinitiontitle
4078   \keys_set:nn { stex / sdefinition }{ #1 }
4079 }
4080
4081 \NewDocumentEnvironment{sdefinition}{0{}}{
4082   \__stex_statements_sdefinition_args:n{ #1 }
4083   \stex_reactivate_macro:N \definiendum
4084   \stex_reactivate_macro:N \definame
4085   \stex_reactivate_macro:N \Definame

```

```

4086 \stex_if_smsmode:F{
4087   \seq_clear:N \l_tmpa_seq
4088   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4089     \str_if_eq:nnF{ ##1 }{}{
4090       \stex_get_symbol:n { ##1 }
4091       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4092         \l_stex_get_symbol_uri_str
4093       }
4094     }
4095   }
4096   \exp_args:Nnnx
4097   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4098   \str_if_empty:NF \sdefinitiontype {
4099     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4100   }
4101   \clist_set:No \l_tmpa_clist \sdefinitiontype
4102   \tl_clear:N \l_tmpa_tl
4103   \clist_map_inline:Nn \l_tmpa_clist {
4104     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4105       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4106     }
4107   }
4108   \tl_if_empty:NTF \l_tmpa_tl {
4109     \__stex_statements_sdefinition_start:
4110   }{
4111     \l_tmpa_tl
4112   }
4113 }
4114 \stex_ref_new_doc_target:n \sdefinitionid
4115 \stex_smsmode_do:
4116 }{
4117   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4118   \stex_if_smsmode:F {
4119     \clist_set:No \l_tmpa_clist \sdefinitiontype
4120     \tl_clear:N \l_tmpa_tl
4121     \clist_map_inline:Nn \l_tmpa_clist {
4122       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4123         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4124       }
4125     }
4126     \tl_if_empty:NTF \l_tmpa_tl {
4127       \__stex_statements_sdefinition_end:
4128     }{
4129       \l_tmpa_tl
4130     }
4131     \end{stex_annotate_env}
4132   }
4133 }

```

\stexpatchdefinition

```

4134 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4135   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4136     ~(\sdefinitiontitle)
4137   }~}

```

```

4138 }
4139 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4140
4141 \newcommand\stexpatchdefinition[3] [] {
4142   \str_set:Nx \l_tmpa_str{ #1 }
4143   \str_if_empty:NTF \l_tmpa_str {
4144     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4145     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4146   }{
4147     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4148     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4149   }
4150 }

```

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

\inlinedef inline:

```

4151 \keys_define:nn {stex / inlinedef }{
4152   type      .str_set_x:N = \sdefinitiontype,
4153   id        .str_set_x:N = \sdefinitionid,
4154   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4155   name      .str_set_x:N = \sdefinitionname
4156 }
4157 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4158   \str_clear:N \sdefinitiontype
4159   \str_clear:N \sdefinitionid
4160   \str_clear:N \sdefinitionname
4161   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4162   \keys_set:nn { stex / inlinedef }{ #1 }
4163 }
4164 \NewDocumentCommand \inlinedef { 0{} m } {
4165   \begin{group}
4166     \__stex_statements_inlinedef_args:n{ #1 }
4167     \stex_reactivate_macro:N \definiendum
4168     \stex_reactivate_macro:N \definame
4169     \stex_reactivate_macro:N \Definame
4170     \stex_ref_new_doc_target:n \sdefinitionid
4171     \stex_if_smsmode:TF{
4172       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4173     }{
4174       \seq_clear:N \l_tmpa_seq
4175       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4176         \str_if_eq:nnF{ ##1 }{}{
4177           \stex_get_symbol:n { ##1 }
4178           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4179             \l_stex_get_symbol_uri_str
4180           }
4181         }
4182       }
4183       \exp_args:Nnx
4184       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4185         \str_if_empty:NF \sdefinitiontype {
4186           \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{
4187

```

```

4188     #2
4189     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4190   }
4191 }
4192 \endgroup
4193 \stex_smsmode_do:
4194 }

```

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

33.2 Assertions

sassertion

```

4195
4196 \keys_define:nn {stex / sassertion }{
4197   type      .str_set_x:N = \sassertiontype,
4198   id        .str_set_x:N = \sassertionid,
4199   title     .tl_set:N    = \sassertiontitle ,
4200   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4201   name      .str_set_x:N = \sassertionname
4202 }
4203 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4204   \str_clear:N \sassertiontype
4205   \str_clear:N \sassertionid
4206   \str_clear:N \sassertionname
4207   \clist_clear:N \l__stex_statements_sassertion_for_clist
4208   \tl_clear:N \sassertiontitle
4209   \keys_set:nn { stex / sassertion }{ #1 }
4210 }
4211
4212 %\tl_new:N \g__stex_statements_aftergroup_tl
4213
4214 \NewDocumentEnvironment{sassertion}{0{}}{
4215   \__stex_statements_sassertion_args:n{ #1 }
4216   \stex_if_smsmode:F {
4217     \seq_clear:N \l_tmpa_seq
4218     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4219       \str_if_eq:nnF{ ##1 }{ }{
4220         \stex_get_symbol:n { ##1 }
4221         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4222           \l_stex_get_symbol_uri_str
4223         }
4224       }
4225     }
4226     \exp_args:Nnnx
4227     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4228     \str_if_empty:NF \sassertiontype {
4229       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4230     }
4231     \clist_set:Nn \l_tmpa_clist \sassertiontype
4232     \tl_clear:N \l_tmpa_tl
4233     \clist_map_inline:Nn \l_tmpa_clist {
4234       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{

```

```

4235         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4236     }
4237 }
4238 \tl_if_empty:NTF \l_tmpa_tl {
4239     \__stex_statements_sassertion_start:
4240 }{
4241     \l_tmpa_tl
4242 }
4243 }
4244 \str_if_empty:NTF \sassertionid {
4245     \str_if_empty:NF \sassertionname {
4246         \stex_ref_new_doc_target:n {}
4247     }
4248 } {
4249     \stex_ref_new_doc_target:n \sassertionid
4250 }
4251 \stex_smsmode_do:
4252 }{
4253     \str_if_empty:NF \sassertionname {
4254         \stex_symdecl_do:nn{ }\sassertionname}
4255         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4256     }
4257     \stex_if_smsmode:F {
4258         \clist_set:No \l_tmpa_clist \sassertiontype
4259         \tl_clear:N \l_tmpa_tl
4260         \clist_map_inline:Nn \l_tmpa_clist {
4261             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4262                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4263             }
4264         }
4265         \tl_if_empty:NTF \l_tmpa_tl {
4266             \__stex_statements_sassertion_end:
4267         }{
4268             \l_tmpa_tl
4269         }
4270         \end{stex_annotate_env}
4271     }
4272 }

```

\stexpatchassertion

```

4273
4274 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4275     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4276         (\sassertiontitle)
4277     }~}
4278 }
4279 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4280
4281 \newcommand\stexpatchassertion[3] [] {
4282     \str_set:Nx \l_tmpa_str{ #1 }
4283     \str_if_empty:NTF \l_tmpa_str {
4284         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4285         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4286     }{

```



```

4287     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4288     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4289   }
4290 }

```

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

`\inlineass` inline:

```

4291 \keys_define:nn {stex / inlineass }{
4292   type      .str_set_x:N = \sassertiontype,
4293   id        .str_set_x:N = \sassertionid,
4294   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4295   name      .str_set_x:N = \sassertionname
4296 }
4297 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4298   \str_clear:N \sassertiontype
4299   \str_clear:N \sassertionid
4300   \str_clear:N \sassertionname
4301   \clist_clear:N \l__stex_statements_sassertion_for_clist
4302   \keys_set:nn { stex / inlineass }{ #1 }
4303 }
4304 \NewDocumentCommand \inlineass { 0{} m } {
4305   \begingroup
4306   \__stex_statements_inlineass_args:n{ #1 }
4307   \str_if_empty:NTF \sassertionid {
4308     \str_if_empty:NF \sassertionname {
4309       \stex_ref_new_doc_target:n {}
4310     }
4311   } {
4312     \stex_ref_new_doc_target:n \sassertionid
4313   }
4314
4315   \stex_if_smsmode:TF{
4316     \str_if_empty:NF \sassertionname {
4317       \stex_symdecl_do:nn{}{\sassertionname}
4318       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4319     }
4320   }{
4321     \seq_clear:N \l_tmpa_seq
4322     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4323       \str_if_eq:nnF{ ##1 }{}{
4324         \stex_get_symbol:n { ##1 }
4325         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4326           \l_stex_get_symbol_uri_str
4327         }
4328       }
4329     }
4330     \exp_args:Nnx
4331     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4332       \str_if_empty:NF \sassertiontype {
4333         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4334       }
4335       #2
4336       \str_if_empty:NF \sassertionname {

```

```

4337     \stex_symdecl_do:nn{}{\sassertionname}
4338     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4339   }
4340 }
4341 }
4342 \endgroup
4343 \stex_smsmode_do:
4344 }

```

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

33.3 Examples

`sexample`

```

4345
4346 \keys_define:nn {stex / sexample }{
4347   type      .str_set_x:N = \exampletype,
4348   id        .str_set_x:N = \sexampleid,
4349   title     .tl_set:N     = \sexamplename,
4350   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4351 }
4352 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4353   \str_clear:N \sexampletype
4354   \str_clear:N \sexampleid
4355   \tl_clear:N \sexamplename
4356   \clist_clear:N \l__stex_statements_sexample_for_clist
4357   \keys_set:nn { stex / sexample }{ #1 }
4358 }
4359
4360 \NewDocumentEnvironment{sexample}{0{}}{
4361   \__stex_statements_sexample_args:n{ #1 }
4362   \stex_if_smsmode:F {
4363     \seq_clear:N \l_tmpa_seq
4364     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4365       \str_if_eq:nnF{ ##1 }{ }{
4366         \stex_get_symbol:n { ##1 }
4367         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4368           \l_stex_get_symbol_uri_str
4369         }
4370       }
4371     }
4372     \exp_args:Nnnx
4373     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4374     \str_if_empty:NF \sexampletype {
4375       \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
4376     }
4377     \clist_set:Nn \l_tmpa_clist \sexampletype
4378     \tl_clear:N \l_tmpa_tl
4379     \clist_map_inline:Nn \l_tmpa_clist {
4380       \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{
4381         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sexample_##1_start:}}
4382       }
4383     }

```

```

4384 \tl_if_empty:NTF \l_tmpa_tl {
4385   \__stex_statements_sexample_start:
4386 }{
4387   \l_tmpa_tl
4388 }
4389 }
4390 \str_if_empty:NF \sexampleid {
4391   \stex_ref_new_doc_target:n \sexampleid
4392 }
4393 \stex_smsmode_do:
4394 }{
4395   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn}{\sexamplename} }
4396   \stex_if_smsmode:F {
4397     \clist_set:Nn \l_tmpa_clist \sexamplename
4398     \tl_clear:N \l_tmpa_tl
4399     \clist_map_inline:Nn \l_tmpa_clist {
4400       \tl_if_exist:cT {\__stex_statements_sexample_##1_end:}{
4401         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sexample_##1_end:}}
4402       }
4403     }
4404     \tl_if_empty:NTF \l_tmpa_tl {
4405       \__stex_statements_sexample_end:
4406     }{
4407       \l_tmpa_tl
4408     }
4409     \end{stex_annotate_env}
4410   }
4411 }

```

`\stexpatchexample`

```

4412
4413 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4414   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4415     (\sexamplename)
4416   }~}
4417 }
4418 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4419
4420 \newcommand\stexpatchexample[3]{} {
4421   \str_set:Nx \l_tmpa_str{ #1 }
4422   \str_if_empty:NTF \l_tmpa_str {
4423     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4424     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4425   }{
4426     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4427     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4428   }
4429 }

```

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

`\inlineex` inline:

```

4430 \keys_define:nn {stex / inlineex }{
4431   type .str_set_x:N = \sexamplename,

```

```

4432 id      .str_set_x:N = \sexampleid,
4433 for      .clist_set:N = \l__stex_statements_sexample_for_clist ,
4434 name     .str_set_x:N = \sexamplename
4435 }
4436 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4437   \str_clear:N \sexamplotype
4438   \str_clear:N \sexampleid
4439   \str_clear:N \sexamplename
4440   \clist_clear:N \l__stex_statements_sexample_for_clist
4441   \keys_set:nn { stex / inlineex }{ #1 }
4442 }
4443 \NewDocumentCommand \inlineex { 0{} m } {
4444   \begingroup
4445     \__stex_statements_inlineex_args:n{ #1 }
4446     \str_if_empty:NF \sexampleid {
4447       \stex_ref_new_doc_target:n \sexampleid
4448     }
4449     \stex_if_smsmode:TF{
4450       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4451   }{
4452     \seq_clear:N \l_tmpa_seq
4453     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4454       \str_if_eq:nnF{ ##1 }{ }{
4455         \stex_get_symbol:n { ##1 }
4456         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4457           \l_stex_get_symbol_uri_str
4458         }
4459       }
4460     }
4461     \exp_args:Nnx
4462     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4463       \str_if_empty:NF \sexamplotype {
4464         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4465       }
4466       #2
4467       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4468   }
4469 }
4470 \endgroup
4471 \stex_smsmode_do:
4472 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4473 \keys_define:nn { stex / sparagraph } {
4474   id      .str_set_x:N = \sparagraphid ,
4475   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4476   type    .str_set_x:N = \sparagraphtype ,
4477   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4478   from    .tl_set:N    = \sparagraphfrom ,

```

```

4479 to .tl_set:N = \sparagraphto ,
4480 start .tl_set:N = \l_stex_sparagraph_start_tl ,
4481 name .str_set:N = \sparagraphname
4482 }
4483
4484 \cs_new_protected:Nn \stex_sparagraph_args:n {
4485 \tl_clear:N \l_stex_sparagraph_title_tl
4486 \tl_clear:N \sparagraphfrom
4487 \tl_clear:N \sparagraphto
4488 \tl_clear:N \l_stex_sparagraph_start_tl
4489 \str_clear:N \sparagraphid
4490 \str_clear:N \sparagraphtype
4491 \clist_clear:N \l__stex_statements_sparagraph_for_clist
4492 \str_clear:N \sparagraphname
4493 \keys_set:nn { stex / sparagraph }{ #1 }
4494 }
4495 \newif\if@in@omtext\@in@omtextfalse
4496
4497 \NewDocumentEnvironment {sparagraph} { 0{} } {
4498 \stex_sparagraph_args:n { #1 }
4499 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4500 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4501 }{
4502 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4503 }
4504 \@in@omtexttrue
4505 \stex_if_smsmode:F {
4506 \seq_clear:N \l_tmpa_seq
4507 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4508 \str_if_eq:nnF{ ##1 }{}{
4509 \stex_get_symbol:n { ##1 }
4510 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4511 \l_stex_get_symbol_uri_str
4512 }
4513 }
4514 }
4515 \exp_args:Nnnx
4516 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4517 \str_if_empty:NF \sparagraphtype {
4518 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4519 }
4520 \str_if_empty:NF \sparagraphfrom {
4521 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4522 }
4523 \str_if_empty:NF \sparagraphto {
4524 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4525 }
4526 \clist_set:N \l_tmpa_clist \sparagraphtype
4527 \tl_clear:N \l_tmpa_tl
4528 \clist_map_inline:Nn \sparagraphtype {
4529 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4530 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4531 }
4532 }

```

```

4533 \tl_if_empty:NTF \l_tmpa_tl {
4534   \__stex_statements_sparagraph_start:
4535 }{
4536   \l_tmpa_tl
4537 }
4538 }
4539 \clist_set:No \l_tmpa_clist \sparagraphtype
4540 \str_if_empty:NTF \sparagraphid {
4541   \str_if_empty:NTF \sparagraphname {
4542     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4543       \stex_ref_new_doc_target:n {}
4544     }
4545   } {
4546     \stex_ref_new_doc_target:n {}
4547   }
4548 } {
4549   \stex_ref_new_doc_target:n \sparagraphid
4550 }
4551 \exp_args:NNx
4552 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4553   \clist_map_inline:Nn \__stex_statements_sparagraph_for_clist {
4554     \str_if_eq:nnF{ ##1 }{ }{
4555       \stex_get_symbol:n { ##1 }
4556       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4557     }
4558   }
4559 }
4560 \stex_smsmode_do:
4561 \ignorespacesandpars
4562 }{
4563   \str_if_empty:NF \sparagraphname {
4564     \stex_symdecl_do:nn{ }\sparagraphname}
4565     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4566   }
4567   \stex_if_smsmode:F {
4568     \clist_set:No \l_tmpa_clist \sparagraphtype
4569     \tl_clear:N \l_tmpa_tl
4570     \clist_map_inline:Nn \l_tmpa_clist {
4571       \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
4572         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
4573       }
4574     }
4575     \tl_if_empty:NTF \l_tmpa_tl {
4576       \__stex_statements_sparagraph_end:
4577     }{
4578       \l_tmpa_tl
4579     }
4580     \end{stex_annotate_env}
4581   }
4582 }

```

\stexpatchparagraph

```

4583
4584 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {

```

```

4585 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4586   \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4587     \titleemph{\l_stex_sparagraph_title_tl}:~
4588   }
4589 }{
4590   \titleemph{\l_stex_sparagraph_start_tl}~
4591 }
4592 }
4593 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4594
4595 \newcommand\stexpatchparagraph[3] [] {
4596   \str_set:Nx \l_tmpa_str{ #1 }
4597   \str_if_empty:NTF \l_tmpa_str {
4598     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4599     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4600   }{
4601     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4602     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4603   }
4604 }
4605
4606 \keys_define:nn { stex / inlinepara } {
4607   id      .str_set:N = \sparagraphid ,
4608   type    .str_set:N = \sparagraphtype ,
4609   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4610   from    .tl_set:N = \sparagraphfrom ,
4611   to      .tl_set:N = \sparagraphto ,
4612   name    .str_set:N = \sparagraphname
4613 }
4614 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4615   \tl_clear:N \sparagraphfrom
4616   \tl_clear:N \sparagraphto
4617   \str_clear:N \sparagraphid
4618   \str_clear:N \sparagraphtype
4619   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4620   \str_clear:N \sparagraphname
4621   \keys_set:nn { stex / inlinepara }{ #1 }
4622 }
4623 \NewDocumentCommand \inlinepara { O{} m } {
4624   \begingroup
4625   \__stex_statements_inlinepara_args:n{ #1 }
4626   \clist_set:No \l_tmpa_clist \sparagraphtype
4627   \str_if_empty:NTF \sparagraphid {
4628     \str_if_empty:NTF \sparagraphname {
4629       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4630         \stex_ref_new_doc_target:n {}
4631       }
4632     } {
4633       \stex_ref_new_doc_target:n {}
4634     }
4635   } {
4636     \stex_ref_new_doc_target:n \sparagraphid
4637   }
4638   \stex_if_smsmode:TF{

```

```

4639 \str_if_empty:NF \sparagraphname {
4640 \stex_symdecl_do:nn{}{\sparagraphname}
4641 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4642 }
4643 }{
4644 \seq_clear:N \l_tmpa_seq
4645 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4646 \str_if_eq:nnF{ ##1 }{}{
4647 \stex_get_symbol:n { ##1 }
4648 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4649 \l_stex_get_symbol_uri_str
4650 }
4651 }
4652 }
4653 \exp_args:Nnx
4654 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4655 \str_if_empty:NF \sparagraphtype {
4656 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}}
4657 }
4658 \str_if_empty:NF \sparagraphfrom {
4659 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
4660 }
4661 \str_if_empty:NF \sparagraphto {
4662 \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
4663 }
4664 \str_if_empty:NF \sparagraphname {
4665 \stex_symdecl_do:nn{}{\sparagraphname}
4666 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4667 }
4668 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4669 \clist_map_inline:Nn \l_tmpa_seq {
4670 \stex_ref_new_sym_target:n {##1}
4671 }
4672 }
4673 #2
4674 }
4675 }
4676 \endgroup
4677 \stex_smsmode_do:
4678 }
4679

```

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

symboldoc

```

4680 \NewDocumentEnvironment{symboldoc}{ m }{
4681 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4682 \seq_clear:N \l_tmpb_seq
4683 \seq_map_inline:Nn \l_tmpa_seq {
4684 \str_if_eq:nnF{ ##1 }{}{
4685 \stex_get_symbol:n { ##1 }
4686 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4687 \l_stex_get_symbol_uri_str
4688 }

```



```

4689     }
4690   }
4691   \par
4692   \exp_args:Nnnx
4693   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4694   ){
4695     \end{stex_annotate_env}
4696   }
4697 </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

```
4698 <*package>
4699 <@@=stex_sproof>
4700
4701 %%%%%%%%%%%%%% sproof.dtx %%%%%%%%%%%%%%
4702
```

34.2 Proofs

We first define some keys for the proof environment.

```
4703 \keys_define:nn { stex / spf } {
4704   id          .str_set:N = \l__stex_sproof_spf_id_str,
4705   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4706   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4707   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4708   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4709   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4710   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4711   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4712   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4713   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4714 }
4715 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4716   \str_clear:N \l__stex_sproof_spf_id_str
4717   \tl_clear:N \l__stex_sproof_spf_display_tl
4718   \tl_clear:N \l__stex_sproof_spf_for_tl
4719   \tl_clear:N \l__stex_sproof_spf_from_tl
4720   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4721   \tl_clear:N \l__stex_sproof_spf_type_tl
4722   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4723 \tl_clear:N \l__stex_sproof_spf_continues_tl
4724 \tl_clear:N \l__stex_sproof_spf_functions_tl
4725 \tl_clear:N \l__stex_sproof_spf_method_tl
4726 \keys_set:nn { stex / spf }{ #1 }
4727 }

```

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

```

4728 \def\spf@flow{flow}

```

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

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, L^AT_EX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

`pst@with@label` This environment manages⁶ the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by T_EX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

4729 \newcount\count_ten
4730 \newenvironment{pst@with@label}[1]{
4731   \edef\pst@label{#1}
4732   \advance\count_ten by 1\relax
4733   \count_ten=1
4734 }{
4735   \advance\count_ten by -1\relax
4736 }

```

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

```

4737 \def\the@pst@label{
4738   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4739 }

```

(End definition for `\the@pst@label`. This function is documented on page ??.)

`\setpstlabelstyle` `\setpstlabelstyle{metaKey-Val pairs}` makes the labeling style customizable. `\setpstlabelstyle{pr}` will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. `\setpstlabelstyledefault` will set the labeling style back to default.

```

4740 \keys_define:nn { stex / pstlabel }{
4741   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4742   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4743   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4744 }
4745 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4746 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4747 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4748 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4749 }
4750 \__stex_sproof_pstlabel_args:n {}
4751 \newcommand\setpstlabelstyle[1]{
4752   \__stex_sproof_pstlabel_args:n {#1}
4753 }
4754 \newcommand\setpstlabelstyledefault{%
4755   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4756 }

```

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

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

```

4757 \ExplSyntaxOff
4758 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4759 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4760 \def\pst@make@label@short#1#2{#2}
4761 \def\pst@make@label@empty#1#2{}
4762 \ExplSyntaxOn
4763 \def\pstlabelstyle#1{%
4764   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4765 }%
4766 \pstlabelstyle{long}%

```

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

\next@pst@label \next@pst@label increments the step label at the current level.

```

4767 \def\next@pst@label{%
4768   \global\advance\count\count10 by 1%
4769 }%

```

(End definition for \next@pst@label. This function is documented on page ??.)

\sproofend This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

4770 \def\sproof@box{
4771   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4772 }
4773 \def\spf@proofend{\sproof@box}
4774 \def\sproofend{
4775   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4776     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4777   }
4778 }
4779 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4780 \def\spf@proofsketch@kw{Proof Sketch}
4781 \def\spf@proof@kw{Proof}
4782 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4783 \AddToHook{begindocument}{
4784   \ltx@ifpackageloaded{babel}{
4785     \makeatletter
4786     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4787     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4788       \input{sproof-ngerman.ldf}
4789     }
4790     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4791       \input{sproof-finnish.ldf}
4792     }
4793     \clist_if_in:NnT \l_tmpa_clist {french}{
4794       \input{sproof-french.ldf}
4795     }
4796     \clist_if_in:NnT \l_tmpa_clist {russian}{
4797       \input{sproof-russian.ldf}
4798     }
4799     \makeatother
4800   }{}
4801 }

```

`spfsketch`

```

4802 \newcommand\spfsketch[2][]{
4803   \__stex_sproof_spf_args:n{#1}
4804   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4805     \titleemph{
4806       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4807         \spf@proofsketch@kw
4808       }{
4809         \l__stex_sproof_spf_type_tl
4810       }
4811     }:
4812   }
4813   {~#2}
4814   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4815   \sproofend
4816 }

```

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

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

```

4817 \newenvironment{spfeq}[2][]{
4818   \__stex_sproof_spf_args:n{#1}
4819   %\sref@target
4820   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4821     \titleemph{
4822       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4823         \spf@proof@kw
4824       }{

```

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

¹⁵EDNOTE: document above

```

4825     \l__stex_sproof_spf_type_tl
4826   }
4827   }:
4828 }
4829 {-#2}
4830 \begin{displaymath}\begin{array}{rcll}
4831 }{
4832 \end{array}\end{displaymath}
4833 }

```

(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.

```

4834 \newenvironment{spf@proof}[2] []{
4835   \l__stex_sproof_spf_args:n{#1}
4836   %\sref@target
4837   \count_ten=10
4838   \par\noindent
4839   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4840     \titleemph{
4841       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4842         \spf@proof@kw
4843       }{
4844         \l__stex_sproof_spf_type_tl
4845       }
4846     }:
4847   }
4848   {-#2}
4849   %\sref@label@id{this \ifx\spf@type@empty\spf@proof@kw\else\spf@type\fi}
4850   \def\pst@label{}
4851   \newcount\pst@count% initialize the labeling mechanism
4852   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4853   }{
4854     \end{pst@with@label}\end{description}
4855   }
4856   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4857   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4858 \newcommand\spfidea[2] []{
4859   \l__stex_sproof_spf_args:n{#1}
4860   \titleemph{
4861     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4862       \l__stex_sproof_spf_type_tl
4863     }:
4864   }-#2
4865   \sproofend
4866 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these

values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

EdN:16

```

spfstep 16
4867 \newenvironment{spfstep}[1][]{
4868   \_stex_sproof_spf_args:n{#1}
4869   \@in@omtexttrue
4870   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4871     \item[\the@pst@label]
4872   }
4873   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4874     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4875   }
4876   %\sref@label{id{\pst@label}
4877   \ignorespacesandpars
4878 }{
4879   \next@pst@label\ignorespacesandpars
4880 }

```

sproofcomment

```

4881 \newenvironment{sproofcomment}[1][]{
4882   \_stex_sproof_spf_args:n{#1}
4883   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4884     \item[\the@pst@label]
4885   }
4886 }{
4887   \next@pst@label
4888 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the `subproof` environment, a new (lower-level) `proproofof` environment is started.

```

4889 \newenvironment{subproof}[2][]{
4890   \_stex_sproof_spf_args:n{#1}
4891   \def\@test{#2}
4892   \ifx\@test\empty\else
4893     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4894       \item[\the@pst@label]
4895     }{#2}
4896   \fi
4897   \begin{pst@with@label}{\pst@label,\number\count_ten}
4898 }{
4899   \end{pst@with@label}\next@pst@label
4900 }

```

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

```

4901 \newenvironment{spfcases}[2][]{
4902   \def\@test{#1}
4903   \ifx\@test\empty
4904     \begin{subproof}[method=by-cases]{#2}

```

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

```

4905 \else
4906   \begin{subproof}[#1,method=by-cases]{#2}
4907 \fi
4908 }{
4909   \end{subproof}
4910 }

```

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

```

4911 \newenvironment{spfcase}[2] [] {
4912   \__stex_sproof_spf_args:n{#1}
4913   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4914     \item[\the@pst@label]
4915   }
4916   \def\@test{#2}
4917   \ifx\@test\@empty
4918   \else
4919     {\titleemph{#2}:~}
4920   \fi
4921   \begin{pst@with@label}{\pst@label,\number\count_ten}
4922 }{
4923   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4924     \sproofend
4925   }
4926   \end{pst@with@label}
4927   \next@pst@label
4928 }

```

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

```

4929 \newcommand\spfcasesketch[3] [] {
4930   \__stex_sproof_spf_args:n{#1}
4931   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4932     \item[\the@pst@label]
4933   }
4934   \def\@test{#2}
4935   \ifx\@test\@empty
4936   \else
4937     {\titleemph{#2}:~}
4938   \fi#3
4939   \next@pst@label
4940 }%

```

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.

```

4941 \keys_define:nn { stex / just }{
4942   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4943   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4944   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4945   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4946 }

```


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

`justification`

4947 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4948 `\newcommand\premise[2] [] {#2}`

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

`\justarg` the `\justarg` macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

4949 `\newcommand\justarg[2] [] {#2}`

4950 `\</package>`

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

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

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

Chapter 35

STEX -Others Implementation

```
4951 <*package>
4952
4953 %%%%%%%%%% others.dtx %%%%%%%%%%
4954
4955 <@@=stex_others>
      Warnings and error messages
4956 % None
\MSC Math subject classifier
4957 \NewDocumentCommand \MSC {m} {
4958 % TODO
4959 }
      (End definition for \MSC. This function is documented on page ??.)
      Patching tikzinput, if loaded
4960 \@ifpackageloaded{tikzinput}{
4961 \RequirePackage{stex-tikzinput}
4962 }{}
4963 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4964 \*package>
4965 \@@=stex_modules>
4966
4967 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4968
4969 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4970 \begingroup
4971 \stex_module_setup:nn{
4972   ns=\c_stex_metatheory_ns_str,
4973   meta=NONE
4974 }{Metatheory}
4975 \stex_reactivate_macro:N \symdecl
4976 \stex_reactivate_macro:N \notation
4977 \stex_reactivate_macro:N \symdef
4978 \ExplSyntaxOff
4979 \csname stex_suppress_html:n\endcsname{
4980   % is-a (a:A, a \in A, a is an A, etc.)
4981   \symdecl[args=ai]{isa}
4982   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4983   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4984   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4985
4986   % bind (\forall, \Pi, \lambda etc.)
4987   \symdecl[args=Bi]{bind}
4988   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4989   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4990   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4991
4992   % dummy variable
4993   \symdecl{dummyvar}
4994   \notation[underscore]{dummyvar}{\comp\_}
4995   \notation[dot]{dummyvar}{\comp\cdot}
4996   \notation[dash]{dummyvar}{\comp{\rm --}}
4997
4998   %fromto (function space, Hom-set, implication etc.)
```

```

4999 \symdecl[args=ai]{fromto}
5000 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
5001 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
5002
5003 % mapto (lambda etc.)
5004 %\symdecl[args=Bi]{mapto}
5005 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
5006 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5007 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5008
5009 % function/operator application
5010 \symdecl[args=ia]{apply}
5011 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5012 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
5013
5014 % ‘type’ of all collections (sets, classes, types, kinds)
5015 \symdecl{collection}
5016 \notation[U]{collection}{\comp{\mathcal{U}}}
5017 \notation[set]{collection}{\comp{\textsf{Set}}}
5018
5019 % collection of propositions/booleans/truth values
5020 \symdecl[name=proposition]{prop}
5021 \notation[prop]{prop}{\comp{\rm prop}}
5022 \notation[BOOL]{prop}{\comp{\rm BOOL}}
5023
5024 % sequences
5025 \symdecl[args=1]{seqtype}
5026 \notation[kleene]{seqtype}{#1^{\comp\ast}}
5027
5028 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
5029 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
5030
5031 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{##1\comp,##2}
5032 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses},#2}{##1\comp,##2}
5033 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
5034
5035 % letin (‘let’, local definitions, variable substitution)
5036 \symdecl[args=bii]{letin}
5037 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=#2}\; \comp{\rm in}}\;#3}
5038 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
5039 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
5040
5041 % structures
5042 \symdecl*[args=1]{module-type}
5043 \notation{module-type}{\mathtt{MOD} #1}
5044 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
5045 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5046
5047 }
5048 \ExplSyntaxOn
5049 \stex_add_to_current_module:n{
5050   \let\appa\apply
5051   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5052   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5053 \def\livar{\csname sequence-index\endcsname[li]}
5054 \def\uivar{\csname sequence-index\endcsname[ui]}
5055 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5056 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5057 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5058 }
5059 \__stex_modules_end_module:
5060 \endgroup
5061 \</package>

```

Chapter 37

Tikzinput Implementation

```
5062 <*package>
5063
5064 %%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%
5065
5066 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5067 \RequirePackage{l3keys2e}
5068
5069 \keys_define:nn { tikzinput } {
5070   image .bool_set:N = \c_tikzinput_image_bool,
5071   image .default:n = false ,
5072   unknown .code:n = {}
5073 }
5074
5075 \ProcessKeysOptions { tikzinput }
5076
5077 \bool_if:NTF \c_tikzinput_image_bool {
5078   \RequirePackage{graphicx}
5079
5080   \providecommand\usetikzlibrary[]{}
5081   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
5082 }{
5083   \RequirePackage{tikz}
5084   \RequirePackage{standalone}
5085
5086   \newcommand \tikzinput [2] [] {
5087     \setkeys{Gin}{#1}
5088     \ifx \Gin@ewidth \Gin@exclamation
5089       \ifx \Gin@eheight \Gin@exclamation
5090         \input { #2 }
5091       \else
5092         \resizebox{!}{ \Gin@eheight }{
5093           \input { #2 }
5094         }
5095       \fi
5096     \else
5097       \ifx \Gin@eheight \Gin@exclamation
5098         \resizebox{ \Gin@ewidth }{!}{
5099           \input { #2 }
```

```

5100     }
5101     \else
5102         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5103             \input { #2 }
5104         }
5105     \fi
5106 \fi
5107 }
5108 }
5109
5110 \newcommand \ctikzinput [2] [] {
5111     \begin{center}
5112         \tikzinput [1] {#2}
5113     \end{center}
5114 }
5115
5116 \@ifpackageloaded{stex}{
5117     \RequirePackage{stex-tikzinput}
5118 }{}
5119
5120 \</package>
5121 \<*stex>
5122 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5123 \RequirePackage{stex}
5124 \RequirePackage{tikzinput}
5125
5126 \newcommand\mhtikzinput [2] [] {%
5127     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5128     \stex_in_repository:nn\Gin@mhrepos{
5129         \tikzinput [1]{\mhpath{##1}{#2}}
5130     }
5131 }
5132 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5133 \</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.

```
5134 \*cls)
5135 \@@=document_structure)
5136 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5137 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

38.2 Class Options

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

`\omdoc@cls@class`

```
5138 \keys_define:nn{ document-structure / pkg }{
5139   class      .str_set_x:N = \c_document_structure_class_str,
5140   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5141   report     .code:n      = {
5142     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5143     \str_set:Nn \c_document_structure_class_str {report}
5144   },
5145   book       .code:n      = {
5146     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5147     \str_set:Nn \c_document_structure_class_str {book}
5148   },
5149   bookpart   .code:n      = {
5150     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5151     \str_set:Nn \c_document_structure_class_str {book}
5152     \str_set:Nn \c_document_structure_topsect_str {chapter}
5153   },
```



```

5154 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5155 unknown     .code:n      = {
5156   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5157 }
5158 }
5159 \ProcessKeysOptions{ document-structure / pkg }
5160 \str_if_empty:NT \c_document_structure_class_str {
5161   \str_set:Nn \c_document_structure_class_str {article}
5162 }
5163 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5164   {\c_document_structure_class_str}
5165

```

38.3 Beefing up the document environment

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

```

5166 \RequirePackage{document-structure}
5167 \bool_if:NF \c_document_structure_minimal_bool {

```

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

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

```

5168 \keys_define:nn { document-structure / document }{
5169   id .str_set_x:N = \c_document_structure_document_id_str
5170 }
5171 \let\__document_structure_orig_document=\document
5172 \renewcommand{\document}[1][]{
5173   \keys_set:nn{ document-structure / document }{ #1 }
5174   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5175   \__document_structure_orig_document
5176 }

```

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

```

5177 }
5178 \</cls>

```

38.4 Implementation: document-structure Package

```

5179 \<*package>
5180 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5181 \RequirePackage{expl-keystr-compat,13keys2e}

```

38.5 Package Options

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

¹⁸EDNOTE: faking documentkeys for now. @HANG, please implement

```

5182
5183 \keys_define:nn{ document-structure / pkg }{
5184   class      .str_set_x:N = \c_document_structure_class_str,
5185   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5186   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5187 }
5188 \ProcessKeysOptions{ document-structure / pkg }
5189 \str_if_empty:NT \c_document_structure_class_str {
5190   \str_set:Nn \c_document_structure_class_str {article}
5191 }
5192 \str_if_empty:NT \c_document_structure_topsect_str {
5193   \str_set:Nn \c_document_structure_topsect_str {section}
5194 }

```

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

```

5195 \RequirePackage{xspace}
5196 \RequirePackage{comment}
5197 \AddToHook{begindocument}{
5198   \ltx@ifpackageloaded{babel}{
5199     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5200     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5201       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5202     }
5203   }{}
5204 }

```

`\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.

```

5205 \int_new:N \l_document_structure_section_level_int
5206 \str_case:VnF \c_document_structure_topsect_str {
5207   {part}}{
5208     \int_set:Nn \l_document_structure_section_level_int {0}
5209   }
5210   {chapter}}{
5211     \int_set:Nn \l_document_structure_section_level_int {1}
5212   }
5213 }{
5214   \str_case:VnF \c_document_structure_class_str {
5215     {book}}{
5216       \int_set:Nn \l_document_structure_section_level_int {0}
5217     }
5218     {report}}{
5219       \int_set:Nn \l_document_structure_section_level_int {0}
5220     }
5221   }{
5222     \int_set:Nn \l_document_structure_section_level_int {2}
5223   }
5224 }

```

38.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. The hierarchy is adjusted automatically according to the \LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁹

EdN:19

```
5225 \def\current@section@level{document}%
5226 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5227 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5228 \cs_new_protected:Npn \skipomgroup {
5229   \ifcase\l_document_structure_section_level_int
5230   \or\stepcounter{part}
5231   \or\stepcounter{chapter}
5232   \or\stepcounter{section}
5233   \or\stepcounter{subsection}
5234   \or\stepcounter{subsubsection}
5235   \or\stepcounter{paragraph}
5236   \or\stepcounter{subparagraph}
5237   \fi
5238 }
```

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

`blindomgroup`

```
5239 \newcommand\at@begin@blindomgroup[1]{%
5240 \newenvironment{blindomgroup}
5241 {
5242   \int_incr:N\l_document_structure_section_level_int
5243   \at@begin@blindomgroup\l_document_structure_section_level_int
5244 }{}}
```

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

```
5245 \newcommand\omgroup@nonum[2]{
5246   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5247   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5248 }
```

(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.

```
5249 \newcommand\omgroup@num[2]{
```

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

```

5250 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5251   \@nameuse{#1}{#2}
5252 }{
5253   \cs_if_exist:NTF\rdfmata@sectioning{
5254     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5255   }{
5256     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5257   }
5258 }
5259 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
5260 }

```

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

omgroup

```

5261 \keys_define:nn { document-structure / omgroup }{
5262   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5263   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5264   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5265   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5266   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5267   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5268   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5269   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5270   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5271   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5272 }
5273 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5274   \str_clear:N \l__document_structure_omgroup_id_str
5275   \str_clear:N \l__document_structure_omgroup_date_str
5276   \clist_clear:N \l__document_structure_omgroup_creators_clist
5277   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5278   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5279   \tl_clear:N \l__document_structure_omgroup_type_tl
5280   \tl_clear:N \l__document_structure_omgroup_short_tl
5281   \tl_clear:N \l__document_structure_omgroup_display_tl
5282   \tl_clear:N \l__document_structure_omgroup_intro_tl
5283   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5284   \keys_set:nn { document-structure / omgroup } { #1 }
5285 }

```

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.

```

5286 \newif\if@mainmatter\@mainmattertrue
5287 \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.

```

5288 \keys_define:nn { document-structure / sectioning }{
5289   name .str_set_x:N = \l__document_structure_sect_name_str ,
5290   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5291   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5292   clear .default:n = {true} ,
5293   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5294   num      .default:n    = {true}
5295 }
5296 \cs_new_protected:Nn \__document_structure_sect_args:n {
5297   \str_clear:N \l__document_structure_sect_name_str
5298   \str_clear:N \l__document_structure_sect_ref_str
5299   \bool_set_false:N \l__document_structure_sect_clear_bool
5300   \bool_set_false:N \l__document_structure_sect_num_bool
5301   \keys_set:nn { document-structure / sectioning } { #1 }
5302 }
5303 \newcommand\omdoc@sectioning[3][]{
5304   \__document_structure_sect_args:n {#1}
5305   \let\omdoc@sect@name\l__document_structure_sect_name_str
5306   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5307   \if@mainmatter% numbering not overridden by frontmatter, etc.
5308     \bool_if:NTF \l__document_structure_sect_num_bool {
5309       \omgroup@num{#2}{#3}
5310     }{
5311       \omgroup@nonum{#2}{#3}
5312     }
5313     \def\current@section@level{\omdoc@sect@name}
5314   \else
5315     \omgroup@nonum{#2}{#3}
5316   \fi
5317 }% 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.

```

5318 \newcommand\omgroup@redefine@addtocontents[1]{%
5319 %\edef\__document_structureimport{#1}%
5320 %\@for\@I:=\__document_structureimport\do{%
5321 %\edef\@path{\csname module@\@I @path\endcsname}%
5322 %\@ifundefined{tf@toc}\relax%
5323 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5324 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5325 %\def\addcontentsline##1##2##3{%
5326 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5327 %\else% hyperref.sty not loaded
5328 %\def\addcontentsline##1##2##3{%
5329 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5330 %\fi
5331 }% 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.

```

5332 \newenvironment{omgroup}[2][]{% keys, title
5333 {
5334   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

5335 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5336   \omgroup@redefine@addtocontents{
5337     \@ifundefined{module@id}\used@modules%

```

```

5338     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5339   }
5340 }

```

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

```

5341 \int_incr:N\l_document_structure_section_level_int
5342 \ifcase\l_document_structure_section_level_int
5343   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5344   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5345   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5346   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5347   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5348   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5349   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5350 \fi
5351 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5352 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5353   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5354 }
5355 }% for customization
5356 {}

```

and finally, we localize the sections

```

5357 \newcommand\omdoc@part@kw{Part}
5358 \newcommand\omdoc@chapter@kw{Chapter}
5359 \newcommand\omdoc@section@kw{Section}
5360 \newcommand\omdoc@subsection@kw{Subsection}
5361 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5362 \newcommand\omdoc@paragraph@kw{paragraph}
5363 \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`

```

5364 \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).

```

5365 \cs_if_exist:NTF\frontmatter{
5366   \let\__document_structure_orig_frontmatter\frontmatter
5367   \let\frontmatter\relax
5368 }{
5369   \tl_set:Nn\__document_structure_orig_frontmatter{
5370     \clearpage
5371     \@mainmatterfalse
5372     \pagenumbering{roman}

```

```

5373 }
5374 }
5375 \cs_if_exist:NTF\backmatter{
5376   \let\__document_structure_orig_backmatter\backmatter
5377   \let\backmatter\relax
5378 }{
5379   \tl_set:Nn\__document_structure_orig_backmatter{
5380     \clearpage
5381     \@mainmatterfalse
5382     \pagenumbering{roman}
5383   }
5384 }

```

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.

```

5385 \newenvironment{frontmatter}{
5386   \__document_structure_orig_frontmatter
5387 }{
5388   \cs_if_exist:NTF\mainmatter{
5389     \mainmatter
5390   }{
5391     \clearpage
5392     \@mainmattertrue
5393     \pagenumbering{arabic}
5394   }
5395 }

```

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

```

5396 \newenvironment{backmatter}{
5397   \__document_structure_orig_backmatter
5398 }{
5399   \cs_if_exist:NTF\mainmatter{
5400     \mainmatter
5401   }{
5402     \clearpage
5403     \@mainmattertrue
5404     \pagenumbering{arabic}
5405   }
5406 }

```

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

```

5407 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5408 \def \c__document_structure_document_str{document}
5409 \newcommand\afterprematurestop{}
5410 \def\prematurestop@endomgroup{
5411   \unless\ifx\@currenvir\c__document_structure_document_str
5412     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5413       \expandafter\prematurestop@endomgroup

```

```

5414 \fi
5415 }
5416 \providecommand\prematurestop{
5417 \message{Stopping~sTeX~processing~prematurely}
5418 \prematurestop@endumgroup
5419 \afterprematurestop
5420 \end{document}
5421 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5422 \RequirePackage{etoolbox}
5423 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5424 \newrobustcmd\useSGvar[1]{%
5425 \@ifundefined{sTeX@Gvar@#1}
5426 {\PackageError{document-structure}
5427 {The sTeX Global variable #1 is undefined}
5428 {set it with \protect\setSGvar}}
5429 \@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.

```

5430 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5431 \@ifundefined{sTeX@Gvar@#1}
5432 {\PackageError{document-structure}
5433 {The sTeX Global variable #1 is undefined}
5434 {set it with \protect\setSGvar}}
5435 {\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.

```
5436 \*cls)
5437 \@@=notesslides)
5438 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5439 \RequirePackage{l3keys2e,expl-keystr-compatible}
5440
5441 \keys_define:nn{notesslides / cls}{
5442   class .code:n = {
5443     \PassOptionsToClass{\CurrentOption}{document-structure}
5444     \str_if_eq:nnT{#1}{book}{
5445       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5446     }
5447     \str_if_eq:nnT{#1}{report}{
5448       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5449     }
5450   },
5451   notes .bool_set:N = \c__notesslides_notes_bool ,
5452   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5453   unknown .code:n = {
5454     \PassOptionsToClass{\CurrentOption}{document-structure}
5455     \PassOptionsToClass{\CurrentOption}{beamer}
5456     \PassOptionsToPackage{\CurrentOption}{notesslides}
5457   }
5458 }
5459 \ProcessKeysOptions{ notesslides / cls }
5460 \bool_if:NTF \c__notesslides_notes_bool {
5461   \PassOptionsToPackage{notes=true}{notesslides}
5462 }{
5463   \PassOptionsToPackage{notes=false}{notesslides}
5464 }
5465 \</cls)
```

now we do the same for the notesslides package.

```

5466 <*package>
5467 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5468 \RequirePackage{l3keys2e,expl-keystr-compat}
5469
5470 \keys_define:nn{notesslides / pkg}{
5471   topsect          .str_set_x:N = \c__notesslides_topsect_str,
5472   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
5473   notes            .bool_set:N = \c__notesslides_notes_bool ,
5474   slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
5475   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
5476   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
5477   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
5478   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
5479   unknown          .code:n       = {
5480     \PassOptionsToClass{\CurrentOption}{stex}
5481     \PassOptionsToClass{\CurrentOption}{tikzinput}
5482   }
5483 }
5484 \ProcessKeysOptions{ notesslides / pkg }
5485 \newif\ifnotes
5486 \bool_if:NTF \c__notesslides_notes_bool {
5487   \notesttrue
5488 }{
5489   \notesfalse
5490 }
5491

```

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

```

5492 \str_if_empty:NTF \c__notesslides_topsect_str {
5493   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5494 }{
5495   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5496 }
5497 </package>

```

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

```

5498 <*cls>
5499 \bool_if:NTF \c__notesslides_notes_bool {
5500   \LoadClass{document-structure}
5501 }{
5502   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5503   \newcounter{Item}
5504   \newcounter{paragraph}
5505   \newcounter{subparagraph}
5506   \newcounter{Hfootnote}
5507   \RequirePackage{document-structure}
5508 }

```

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

```

5509 \RequirePackage{notesslides}
5510 </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.

```

5511 \*package>
5512 \bool_if:NT \c__notesslides_notes_bool {
5513   \RequirePackage{a4wide}
5514   \RequirePackage{marginnote}
5515   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5516   \RequirePackage{mdframed}
5517   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5518   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5519 }
5520 \RequirePackage{stex-tikzinput}
5521 \RequirePackage{etoolbox}
5522 \RequirePackage{amssymb}
5523 \RequirePackage{amsmath}
5524 \RequirePackage{comment}
5525 \RequirePackage{textcomp}
5526 \RequirePackage{url}
5527 \RequirePackage{graphicx}
5528 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.²⁰

```

5529 \bool_if:NT \c__notesslides_notes_bool {
5530   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5531 }

```

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

```

5532 \newcounter{slide}
5533 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5534 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

5535 \bool_if:NTF \c__notesslides_notes_bool {
5536   \renewenvironment{note}{\ignorespaces}{}
5537 }{
5538   \excludecomment{note}
5539 }

```

²⁰EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
5540 \bool_if:NT \c__notesslides_notes_bool {
5541   \newlength{\slideframewidth}
5542   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5543 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5544   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5545     \bool_set_true:N #1
5546   }{
5547     \bool_set_false:N #1
5548   }
5549 }
5550 \keys_define:nn{notesslides / frame}{
5551   label .str_set_x:N = \l__notesslides_frame_label_str,
5552   allowframebreaks .code:n = {
5553     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5554   },
5555   allowdisplaybreaks .code:n = {
5556     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5557   },
5558   fragile .code:n = {
5559     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5560   },
5561   shrink .code:n = {
5562     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5563   },
5564   squeeze .code:n = {
5565     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5566   },
5567   t .code:n = {
5568     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5569   },
5570 }
5571 \cs_new_protected:Nn \__notesslides_frame_args:n {
5572   \str_clear:N \l__notesslides_frame_label_str
5573   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5574   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5575   \bool_set_true:N \l__notesslides_frame_fragile_bool
5576   \bool_set_true:N \l__notesslides_frame_shrink_bool
5577   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5578   \bool_set_true:N \l__notesslides_frame_t_bool
5579   \keys_set:nn { notesslides / frame }{ #1 }
5580 }
```

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

```
5581 \renewenvironment{frame}[1][]{
5582   \__notesslides_frame_args:n{#1}
5583   \sffamily
5584   \stepcounter{slide}
5585   \def\@currentlabel{\theslide}
5586   \str_if_empty:NF \l__notesslides_frame_label_str {
5587     \label{\l__notesslides_frame_label_str}
```

5588 }
5589

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

5589 \def\itemize@level{outer}
5590 \def\itemize@outer{outer}
5591 \def\itemize@inner{inner}
5592 \renewcommand\newpage{\addtocounter{framenum}{1}}
5593 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5594 \renewenvironment{itemize}{
5595 \ifx\itemize@level\itemize@outer
5596 \def\itemize@label{\$\rhd\$}
5597 \fi
5598 \ifx\itemize@level\itemize@inner
5599 \def\itemize@label{\$\scriptstyle\rhd\$}
5600 \fi
5601 \begin{list}
5602 {\itemize@label}
5603 {\setlength{\labelsep}{.3em}
5604 \setlength{\labelwidth}{.5em}
5605 \setlength{\leftmargin}{1.5em}
5606 }
5607 \edef\itemize@level{\itemize@inner}
5608 }{
5609 \end{list}
5610 }

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

5611 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5612 }{
5613 \medskip\miko@slidelabel\end{mdframed}
5614 }

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

\frametitle

5615 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5616 }

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

EdN:21

\pause 21

5617 \bool_if:NT \c__notesslides_notes_bool {
5618 \newcommand\pause{
5619 }
5620 }

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

nparagraph

5620 \bool_if:NTF \c__notesslides_notes_bool {
5621 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5622 }{
5623 \excludecomment{nparagraph}
5624 }
5625 }

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

```

nomgroup
5625 \bool_if:NTF \c__notesslides_notes_bool {
5626   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5627 }{
5628   \excludecomment{nomgroup}
5629 }

ndefinition
5630 \bool_if:NTF \c__notesslides_notes_bool {
5631   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5632 }{
5633   \excludecomment{ndefinition}
5634 }

nassertion
5635 \bool_if:NTF \c__notesslides_notes_bool {
5636   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5637 }{
5638   \excludecomment{nassertion}
5639 }

nsproof
5640 \bool_if:NTF \c__notesslides_notes_bool {
5641   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5642 }{
5643   \excludecomment{nproof}
5644 }

nexample
5645 \bool_if:NTF \c__notesslides_notes_bool {
5646   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5647 }{
5648   \excludecomment{nexample}
5649 }

\inputref@*skip We customize the hooks for in \inputref.
5650 \def\inputref@preskip{\smallskip}
5651 \def\inputref@postskip{\medskip}

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

\inputref*
5652 \let\orig@inputref\inputref
5653 \def\inputref{\@ifstar\ninputref\orig@inputref}
5654 \newcommand\ninputref[2] [] {
5655   \bool_if:NT \c__notesslides_notes_bool {
5656     \orig@inputref[#1]{#2}
5657   }
5658 }

(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>}`.

```
5659 \newlength{\slidelogoheight}
5660
5661 \bool_if:NTF \c__notesslides_notes_bool {
5662   \setlength{\slidelogoheight}{.4cm}
5663 }{
5664   \setlength{\slidelogoheight}{1cm}
5665 }
5666 \newsavebox{\slidelogo}
5667 \sbox{\slidelogo}{\TeX}
5668 \newrobustcmd{\setslidelogo}[1]{
5669   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5670 }
```

(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.

```
5671 \def\source{Michael Kohlhase}% customize locally
5672 \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.

```
5673 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5674 \newsavebox{\cclogo}
5675 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5676 \newif\ifcchref\cchreffalse
5677 \AtBeginDocument{
5678   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5679 }
5680 \def\licensing{
5681   \ifcchref
5682     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5683   \else
5684     {\usebox{\cclogo}}
5685   \fi
5686 }
5687 \newrobustcmd{\setlicensing}[2][]{
5688   \def\@url{#1}
5689   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5690   \ifx\@url\@empty
5691     \def\licensing{{\usebox{\cclogo}}}
5692   \else
5693     \def\licensing{
```

```

5694     \ifcchref
5695     \href{#1}{\usebox{\cclogo}}
5696   \else
5697     {\usebox{\cclogo}}
5698   \fi
5699 }
5700 \fi
5701 }

```

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

EdN:22

`\slidelabel` Now, we set up the slide label for the article mode.²²

```

5702 \newrobustcmd\miko@slidelabel{
5703   \vbox to \slidelogoheight{
5704     \vss\hbox to \slidewidth
5705     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5706   }
5707 }

```

(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.

```

5708 \def\Gin@mhrepos{}
5709 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5710 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5711 \newrobustcmd\frameimage[2][]{
5712   \stepcounter{slide}
5713   \bool_if:NT \c__notesslides_frameimages_bool {
5714     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5715     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5716     \begin{center}
5717       \bool_if:NTF \c__notesslides_fiboxed_bool {
5718         \fbox{
5719           \ifx\Gin@ewidth\@empty
5720             \ifx\Gin@mhrepos\@empty
5721               \mhgraphics[width=\slidewidth,#1]{#2}
5722             \else
5723               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5724             \fi
5725           \else% Gin@ewidth empty
5726             \ifx\Gin@mhrepos\@empty
5727               \mhgraphics[#1]{#2}
5728             \else
5729               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5730             \fi
5731           \fi% Gin@ewidth empty
5732         }
5733       }{
5734         \ifx\Gin@ewidth\@empty

```

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


```

5735         \ifx\Gin@mhrepos\empty
5736             \mhgraphics[width=\slidewidth,#1]{#2}
5737         \else
5738             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5739         \fi
5740         \ifx\Gin@mhrepos\empty
5741             \mhgraphics[#1]{#2}
5742         \else
5743             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5744         \fi
5745     \fi% Gin@ewidth empty
5746 }
5747 \end{center}
5748 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5749 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5750 }
5751 } % 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.

```

5752 \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

```

5753 \AddToHook{begindocument}{
5754     \definecolor{green}{rgb}{0,.5,0}
5755     \definecolor{purple}{cmyk}{.3,1,0,.17}
5756 }

```

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`.

```

5757 % \def\STpresent#1{\textcolor{blue}{#1}}
5758 \def\defemph#1{\textcolor{magenta}{#1}}
5759 \def\symrefemph#1{\textcolor{cyan}{#1}}
5760 \def\compemph#1{\textcolor{blue}{#1}}
5761 \def\titleemph#1{\textcolor{blue}{#1}}
5762 \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.

```

5763 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5764 \def\smalltextwarning{
5765     \pgfuseimage{miko@small@dbend}
5766     \xspace
5767 }
5768 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5769 \newrobustcmd\textwarning{
5770   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5771   \xspace
5772 }
5773 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5774 \newrobustcmd\bigtextwarning{
5775   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5776   \xspace
5777 }

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

5778 \newrobustcmd\putgraphicsat[3]{
5779   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5780 }
5781 \newrobustcmd\putat[2]{
5782   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5783 }

```

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`.

```

5784 \bool_if:NT \c__notesslides_sectocframes_bool {
5785   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5786     \newcounter{chapter}\counterwithin*{section}{chapter}
5787   }{
5788     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5789       \newcounter{chapter}\counterwithin*{section}{chapter}
5790     }
5791   }
5792 }

```

`\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

5793 \def\part@prefix{}
5794 \@ifpackageloaded{document-structure}{
5795   \str_case:VnF \__notesslidesstopsect {
5796     {part}{
5797       \int_set:Nn \l_document_structure_section_level_int {0}
5798       \def\thesection{\arabic{chapter}.\arabic{section}}
5799       \def\part@prefix{\arabic{chapter}.}
5800     }
5801     {chapter}{
5802       \int_set:Nn \l_document_structure_section_level_int {1}
5803       \def\thesection{\arabic{chapter}.\arabic{section}}
5804       \def\part@prefix{\arabic{chapter}.}
5805     }
5806   }{
5807     \int_set:Nn \l_document_structure_section_level_int {2}
5808     \def\part@prefix{}

```

```

5809 }
5810 }
5811
5812 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the omgroup environment that choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

5813 \renewenvironment{omgroup}[2][]{
5814   \__document_structure_omgroup_args:n { #1 }
5815   \int_incr:N \l_document_structure_section_level_int
5816   \bool_if:NT \c__notesslides_sectocframes_bool {
5817     \stepcounter{slide}
5818     \begin{frame}[noframenumbering]
5819       \vfill\Large\centering
5820       \red{
5821         \ifcase\l_document_structure_section_level_int\or
5822           \stepcounter{part}
5823           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5824           \def\currentsectionlevel{\omdoc@part@kw}
5825         \or
5826           \stepcounter{chapter}
5827           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5828           \def\currentsectionlevel{\omdoc@chapter@kw}
5829         \or
5830           \stepcounter{section}
5831           \def\__notesslideslabel{\part@prefix\arabic{section}}
5832           \def\currentsectionlevel{\omdoc@section@kw}
5833         \or
5834           \stepcounter{subsection}
5835           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5836           \def\currentsectionlevel{\omdoc@subsection@kw}
5837         \or
5838           \stepcounter{subsubsection}
5839           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5840           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5841         \or
5842           \stepcounter{paragraph}
5843           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5844           \def\currentsectionlevel{\omdoc@paragraph@kw}
5845         \else
5846           \def\__notesslideslabel{}
5847           \def\currentsectionlevel{\omdoc@paragraph@kw}
5848         \fi% end ifcase
5849         \__notesslideslabel%\sref@label@id\__notesslideslabel
5850         \quad #2%
5851       }%
5852     \vfill%
5853   \end{frame}%
5854 }
5855 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5856   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str

```

```

5857     }
5858   }{}
5859 }

```

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

```

5860 \def\inserttheorembodyfont{\normalfont}
5861 %\bool_if:NF \c__notesslides_notes_bool {
5862 %   \defbeamertemplate{theorem begin}{miko}
5863 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5864 %     \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
5865 %     \inserttheorempunctuation\inserttheorembodyfont\xspace}
5866 %   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5867 % \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.

```

5868 % \expandafter\def\csname Parent2\endcsname{}
5869 %}
5870
5871 \AddToHook{begindocument}{ % this does not work for some reason
5872   \setbeamertemplate{theorems}[ams style]
5873 }
5874 \bool_if:NT \c__notesslides_notes_bool {
5875   \renewenvironment{columns}[1][]{%
5876     \par\noindent%
5877     \begin{minipage}%
5878       \linewidth\centering\leavevmode%
5879   }{%
5880     \end{minipage}\par\noindent%
5881   }%
5882   \newsavebox\columnbox%
5883   \renewenvironment<>{column}[2][]{%
5884     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5885   }{%
5886     \end{minipage}\end{lrbox}\usebox\columnbox%
5887   }%
5888 }
5889 \bool_if:NTF \c__notesslides_noproblems_bool {
5890   \newenvironment{problems}{}{}
5891 }{
5892   \excludacomment{problems}
5893 }

```

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.

```

5894 \gdef\printexcursions{}
5895 \newcommand\excursionref[2]{% label, text

```

```

5896 \bool_if:NT \c__notesslides_notes_bool {
5897   \begin{sparagraph}[title=Excursion]
5898     #2 \sref[fallback=the appendix]{#1}.
5899   \end{sparagraph}
5900 }
5901 }
5902 \newcommand\activate@excursion[2][]{
5903   \gappto\printexcursions{\inputref{#1}{#2}}
5904 }
5905 \newcommand\excursion[4][]{% repos, label, path, text
5906   \bool_if:NT \c__notesslides_notes_bool {
5907     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5908   }
5909 }

```

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

\excursiongroup

```

5910 \keys_define:nn{notesslides / excursiongroup }{
5911   id          .str_set_x:N = \l__notesslides_excursion_id_str,
5912   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
5913   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5914 }
5915 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5916   \tl_clear:N \l__notesslides_excursion_intro_tl
5917   \str_clear:N \l__notesslides_excursion_id_str
5918   \str_clear:N \l__notesslides_excursion_mhrepos_str
5919   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5920 }
5921 \newcommand\excursiongroup[1][]{
5922   \__notesslides_excursion_args:n{ #1 }
5923   \ifdefempty\printexcursions{}% only if there are excursions
5924   {\begin{note}
5925     \begin{omgroup}[#1]{Excursions}%
5926     \ifdefempty\l__notesslides_excursion_intro_tl{
5927       \inputref[\l__notesslides_excursion_mhrepos_str]{
5928         \l__notesslides_excursion_intro_tl
5929       }
5930     }
5931     \printexcursions%
5932     \end{omgroup}
5933   \end{note}}
5934 }
5935 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5936 \</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.

```
5937 <*package>
5938 <@@=problems>
5939 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5940 \RequirePackage{l3keys2e,expl-keystr-compatible}
5941
5942 \keys_define:nn { problem / pkg }{
5943   notes      .default:n    = { true },
5944   notes      .bool_set:N   = \c__problems_notes_bool,
5945   gnotes     .default:n    = { true },
5946   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5947   hints      .default:n    = { true },
5948   hints      .bool_set:N   = \c__problems_hints_bool,
5949   solutions  .default:n    = { true },
5950   solutions  .bool_set:N   = \c__problems_solutions_bool,
5951   pts        .default:n    = { true },
5952   pts        .bool_set:N   = \c__problems_pts_bool,
5953   min        .default:n    = { true },
5954   min        .bool_set:N   = \c__problems_min_bool,
5955   boxed      .default:n    = { true },
5956   boxed      .bool_set:N   = \c__problems_boxed_bool,
5957   unknown    .code:n       = {}
5958 }
5959 \newif\ifsolutions
5960
5961 \ProcessKeysOptions{ problem / pkg }
5962 \bool_if:NTF \c__problems_solutions_bool {
5963   \solutionstrue
5964 }{
5965   \solutionsfalse
5966 }
```

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

```
5967 \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.

```
5968 \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.

```
5969 \def\prob@problem@kw{Problem}
5970 \def\prob@solution@kw{Solution}
5971 \def\prob@hint@kw{Hint}
5972 \def\prob@note@kw{Note}
5973 \def\prob@gnote@kw{Grading}
5974 \def\prob@pt@kw{pt}
5975 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5976 \AddToHook{begindocument}{
5977   \ltx@ifpackageloaded{babel}{
5978     \makeatletter
5979     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5980     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5981       \input{problem-ngerman.ldf}
5982     }
5983     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5984       \input{problem-finnish.ldf}
5985     }
5986     \clist_if_in:NnT \l_tmpa_clist {french}{
5987       \input{problem-french.ldf}
5988     }
5989     \clist_if_in:NnT \l_tmpa_clist {russian}{
5990       \input{problem-russian.ldf}
5991     }
5992     \makeatother
5993   }{ }
5994 }
```

40.2 Problems and Solutions

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

```
5995 \keys_define:nn{ problem / problem }{
5996   id      .str_set_x:N = \l__problems_prob_id_str,
5997   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5998   min     .tl_set:N    = \l__problems_prob_min_tl,
5999   title   .tl_set:N    = \l__problems_prob_title_tl,
6000   type    .tl_set:N    = \l__problems_prob_type_tl,
6001   refnum  .int_set:N   = \l__problems_prob_refnum_int
6002 }
6003 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6004 \str_clear:N \l__problems_prob_id_str
6005 \tl_clear:N \l__problems_prob_pts_tl
6006 \tl_clear:N \l__problems_prob_min_tl
6007 \tl_clear:N \l__problems_prob_title_tl
6008 \tl_clear:N \l__problems_prob_type_tl
6009 \int_zero_new:N \l__problems_prob_refnum_int
6010 \keys_set:nn { problem / problem }{ #1 }
6011 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6012   \let\l__problems_prob_refnum_int\undefined
6013 }
6014 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6015 \newcounter{problem}
6016 \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.

```

6017 \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

```

6018 \newcommand\prob@number{
6019   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6020     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6021   }{
6022     \int_if_exist:NTF \l__problems_prob_refnum_int {
6023       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6024     }{
6025       \prob@label\theproblem
6026     }
6027   }
6028 }

```

(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.

```

6029 \newcommand\prob@title[3]{%
6030   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6031     #2 \l__problems_inclprob_title_tl #3
6032   }{
6033     \tl_if_exist:NTF \l__problems_prob_title_tl {
6034       #2 \l__problems_prob_title_tl #3
6035     }{
6036       #1
6037     }
6038   }
6039 }

```


(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.

```
6040 \def\prob@heading{
6041   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6042   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6043 }
```

(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`

```
6044 \newenvironment{sproblem}[1][{}]{
6045   \__problems_prob_args:n{#1}%\sref@target%
6046   \@in@omtexttrue% we are in a statement (for inline definitions)
6047   \stepcounter{problem}\record@problem
6048   \def\current@section@level{\prob@problem@kw}
6049   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6050     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6051   }{
6052     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6053   }
6054   \str_if_exist:NTF \l__problems_inclprob_id_str {
6055     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6056   }{
6057     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6058   }
6059
6060
6061   \clist_set:No \l_tmpa_clist \sproblemtype
6062   \tl_clear:N \l_tmpa_tl
6063   \clist_map_inline:Nn \l_tmpa_clist {
6064     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6065       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6066     }
6067   }
6068   \tl_if_empty:NTF \l_tmpa_tl {
6069     \__problems_sproblem_start:
6070   }{
6071     \l_tmpa_tl
6072   }
6073   \stex_ref_new_doc_target:n \sproblemid
6074 }{
6075   \clist_set:No \l_tmpa_clist \sproblemtype
6076   \tl_clear:N \l_tmpa_tl
6077   \clist_map_inline:Nn \l_tmpa_clist {
6078     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6079       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6080     }
6081   }
```

```

6081 }
6082 \tl_if_empty:NTF \l_tmpa_tl {
6083   \__problems_sproblem_end:
6084 }{
6085   \l_tmpa_tl
6086 }
6087
6088
6089 \smallskip
6090 }
6091
6092
6093 \cs_new_protected:Nn \__problems_sproblem_start: {
6094   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6095 }
6096 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6097
6098 \newcommand\stexpatchproblem[3]{} {
6099   \str_set:Nx \l_tmpa_str{ #1 }
6100   \str_if_empty:NTF \l_tmpa_str {
6101     \tl_set:Nn \__problems_sproblem_start: { #2 }
6102     \tl_set:Nn \__problems_sproblem_end: { #3 }
6103   }{
6104     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6105     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6106   }
6107 }
6108
6109
6110 \bool_if:NT \c__problems_boxed_bool {
6111   \surroundwithmdframed{problem}
6112 }

```

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

```

6113 \def\record@problem{
6114   \protected@write\@auxout{}
6115   {
6116     \string\@problem{\prob@number}
6117     {
6118       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6119         \l__problems_inclprob_pts_tl
6120       }{
6121         \l__problems_prob_pts_tl
6122       }
6123     }%
6124     {
6125       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6126         \l__problems_inclprob_min_tl
6127       }{
6128         \l__problems_prob_min_tl
6129       }
6130     }
6131   }
6132 }

```

(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).

```
6133 \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.

```
6134 \keys_define:nn { problem / solution }{
6135   id                .str_set_x:N = \l__problems_solution_id_str ,
6136   for               .tl_set:N   = \l__problems_solution_for_tl ,
6137   height            .dim_set:N   = \l__problems_solution_height_dim ,
6138   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6139   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
6140   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6141 }
6142 \cs_new_protected:Nn \__problems_solution_args:n {
6143   \str_clear:N \l__problems_solution_id_str
6144   \tl_clear:N \l__problems_solution_for_tl
6145   \tl_clear:N \l__problems_solution_srccite_tl
6146   \clist_clear:N \l__problems_solution_creators_clist
6147   \clist_clear:N \l__problems_solution_contributors_clist
6148   \dim_zero:N \l__problems_solution_height_dim
6149   \keys_set:nn { problem / solution }{ #1 }
6150 }
```

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

```
6151 \newcommand\@startsolution[1][{}]{
6152   \__problems_solution_args:n { #1 }
6153   \@in@omtexttrue% we are in a statement.
6154   \bool_if:NF \c__problems_boxed_bool { \hrule }
6155   \smallskip\noindent
6156   {\textbf\prob@solution@kw : \enspace}
6157   \begin{small}
6158   \def\current@section@level{\prob@solution@kw}
6159   \ignorespacesandpars
6160 }
```

\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.

```
6161 \newcommand\startsolutions{
6162   \specialcomment{solution}{\@startsolution}{
6163     \bool_if:NF \c__problems_boxed_bool {
6164       \hrule\medskip
6165     }
6166     \end{small}%
6167   }
6168   \bool_if:NT \c__problems_boxed_bool {
6169     \surroundwithmdframed{solution}
6170   }
6171 }
```

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

\stopsolutions

```
6172 \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.

```
6173 \ifsolutions
```

```
6174 \startsolutions
```

```
6175 \else
```

```
6176 \stopsolutions
```

```
6177 \fi
```

exnote

```
6178 \bool_if:NTF \c__problems_notes_bool {
```

```
6179 \newenvironment{exnote}[1][]{
```

```
6180 \par\smallskip\hrule\smallskip
```

```
6181 \noindent\textbf{\prob@note@kw : }\small
```

```
6182 }{
```

```
6183 \smallskip\hrule
```

```
6184 }
```

```
6185 }{
```

```
6186 \excludecomment{exnote}
```

```
6187 }
```

hint

```
6188 \bool_if:NTF \c__problems_notes_bool {
```

```
6189 \newenvironment{hint}[1][]{
```

```
6190 \par\smallskip\hrule\smallskip
```

```
6191 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6192 }{
```

```
6193 \smallskip\hrule
```

```
6194 }
```

```
6195 \newenvironment{exhint}[1][]{
```

```
6196 \par\smallskip\hrule\smallskip
```

```
6197 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6198 }{
```

```
6199 \smallskip\hrule
```

```
6200 }
```

```
6201 }{
```

```
6202 \excludecomment{hint}
```

```
6203 \excludecomment{exhint}
```

```
6204 }
```

gnote

```
6205 \bool_if:NTF \c__problems_notes_bool {
```

```
6206 \newenvironment{gnote}[1][]{
```

```
6207 \par\smallskip\hrule\smallskip
```

```
6208 \noindent\textbf{\prob@gnote@kw : }\small
```

```
6209 }{
```

```
6210 \smallskip\hrule
```

```
6211 }
```

```
6212 }{
```

```
6213 \excludecomment{gnote}
```

```
6214 }
```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6215 \newenvironment{mcb}{
6216   \begin{enumerate}
6217 }{
6218   \end{enumerate}
6219 }
```

we define the keys for the mcc macro

```
6220 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6221   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6222     \bool_set_true:N #1
6223   }{
6224     \bool_set_false:N #1
6225   }
6226 }
6227 \keys_define:nn { problem / mcc }{
6228   id          .str_set:N = \l__problems_mcc_id_str ,
6229   feedback    .tl_set:N   = \l__problems_mcc_feedback_tl ,
6230   T           .default:n   = { true } ,
6231   T           .bool_set:N  = \l__problems_mcc_t_bool ,
6232   F           .default:n   = { true } ,
6233   F           .bool_set:N  = \l__problems_mcc_f_bool ,
6234   Ttext       .code:n      = {
6235     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6236   } ,
6237   Ftext       .code:n      = {
6238     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6239   }
6240 }
6241 \cs_new_protected:Nn \l__problems_mcc_args:n {
6242   \str_clear:N \l__problems_mcc_id_str
6243   \tl_clear:N \l__problems_mcc_feedback_tl
6244   \bool_set_true:N \l__problems_mcc_t_bool
6245   \bool_set_true:N \l__problems_mcc_f_bool
6246   \bool_set_true:N \l__problems_mcc_Ttext_bool
6247   \bool_set_false:N \l__problems_mcc_Ftext_bool
6248   \keys_set:nn { problem / mcc }{ #1 }
6249 }
```

\mcc

```
6250 \newcommand\mcc[2][] {
6251   \l__problems_mcc_args:n{ #1 }
6252   \item #2
6253   \ifsolutions
6254     \\\
6255     \bool_if:NT \l__problems_mcc_t_bool {
6256       % TODO!
6257       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6258     }
6259     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

6260         % TODO!
6261         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6262     }
6263     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6264         !
6265     }{
6266         \l__problems_mcc_feedback_tl
6267     }
6268     \fi
6269 } %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.

```

6270
6271 \keys_define:nn{ problem / inclproblem }{
6272   id      .str_set:N = \l__problems_inclprob_id_str,
6273   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6274   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6275   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6276   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6277   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6278   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6279 }
6280 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6281   \str_clear:N \l__problems_prob_id_str
6282   \tl_clear:N \l__problems_inclprob_pts_tl
6283   \tl_clear:N \l__problems_inclprob_min_tl
6284   \tl_clear:N \l__problems_inclprob_title_tl
6285   \tl_clear:N \l__problems_inclprob_type_tl
6286   \int_zero_new:N \l__problems_inclprob_refnum_int
6287   \str_clear:N \l__problems_inclprob_mhrepos_str
6288   \keys_set:nn { problem / inclproblem }{ #1 }
6289   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6290     \let\l__problems_inclprob_pts_tl\undefined
6291   }
6292   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6293     \let\l__problems_inclprob_min_tl\undefined
6294   }
6295   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6296     \let\l__problems_inclprob_title_tl\undefined
6297   }
6298   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6299     \let\l__problems_inclprob_type_tl\undefined
6300   }
6301   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6302     \let\l__problems_inclprob_refnum_int\undefined
6303   }
6304 }

```

```

6305
6306 \cs_new_protected:Nn \__problems_inclprob_clear: {
6307   \let\l__problems_inclprob_id_str\undefined
6308   \let\l__problems_inclprob_pts_tl\undefined
6309   \let\l__problems_inclprob_min_tl\undefined
6310   \let\l__problems_inclprob_title_tl\undefined
6311   \let\l__problems_inclprob_type_tl\undefined
6312   \let\l__problems_inclprob_refnum_int\undefined
6313   \let\l__problems_inclprob_mhrepos_str\undefined
6314 }
6315 \__problems_inclprob_clear:
6316
6317 \newcommand\includeproblem[2][ ]{
6318   \__problems_inclprob_args:n{ #1 }
6319   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6320     \input{#2}
6321   }{
6322     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6323       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6324     }
6325   }
6326   \__problems_inclprob_clear:
6327 }

```

(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.

```

6328 \AddToHook{enddocument}{
6329   \bool_if:NT \c__problems_pts_bool {
6330     \message{Total:~\arabic{pts}~points}
6331   }
6332   \bool_if:NT \c__problems_min_bool {
6333     \message{Total:~\arabic{min}~minutes}
6334   }
6335 }

```

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

```

6336 \def\pts#1{
6337   \bool_if:NT \c__problems_pts_bool {
6338     \marginpar{#1~\prob@pt@kw}
6339   }
6340 }
6341 \def\min#1{
6342   \bool_if:NT \c__problems_min_bool {
6343     \marginpar{#1~\prob@min@kw}
6344   }
6345 }

```

`\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.

```

6346 \newcounter{pts}
6347 \def\show@pts{
6348   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6349     \bool_if:NT \c__problems_pts_bool {
6350       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6351       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6352     }
6353   }{
6354     \tl_if_exist:NT \l__problems_prob_pts_tl {
6355       \bool_if:NT \c__problems_pts_bool {
6356         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6357         \addtocounter{pts}{\l__problems_prob_pts_tl}
6358       }
6359     }
6360   }
6361 }

```

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

`\show@min`

```

6362 \newcounter{min}
6363 \def\show@min{
6364   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6365     \bool_if:NT \c__problems_min_bool {
6366       \marginpar{\l__problems_inclprob_min_tl\ min}
6367       \addtocounter{min}{\l__problems_inclprob_min_tl}
6368     }
6369   }{
6370     \tl_if_exist:NT \l__problems_prob_min_tl {
6371       \bool_if:NT \c__problems_min_bool {
6372         \marginpar{\l__problems_prob_min_tl\ min}
6373         \addtocounter{min}{\l__problems_prob_min_tl}
6374       }
6375     }
6376   }
6377 }
6378 </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.

```
6379 <@@=hwexam>
6380 <*cls>
6381 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6382 \RequirePackage{l3keys2e,expl-keystr-compatible}
6383 \DeclareOption*{
6384   \PassOptionsToClass{\CurrentOption}{document-structure}
6385   \PassOptionsToPackage{\CurrentOption}{stex}
6386   \PassOptionsToPackage{\CurrentOption}{hwexam}
6387   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6388 }
6389 \ProcessOptions
```

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

```
6390 \LoadClass{document-structure}
6391 \RequirePackage{stex}
6392 \RequirePackage{hwexam}
6393 \RequirePackage{tikzinput}
6394 \RequirePackage{graphicx}
6395 \RequirePackage{a4wide}
6396 \RequirePackage{amssymb}
6397 \RequirePackage{amstext}
6398 \RequirePackage{amsmath}
```

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

```

6399 \newcommand\assig@default@type{\hwexam@assignment@kw}
6400 \def\document@hwexamtype{\assig@default@type}
6401 <@@=document_structure>
6402 \keys_define:nn { document-structure / document }{
6403 id .str_set_x:N = \c_document_structure_document_id_str,
6404 hwexamtype .tl_set:N = \document@hwexamtype
6405 }
6406 <@@=hwexam>
6407 </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.

```
6408 \*package>
6409 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6410 \RequirePackage{l3keys2e,expl-keystr-compat}
6411
6412 \newif\iftest\testfalse
6413 \DeclareOption{test}{\testtrue}
6414 \newif\ifmultiple\multiplefalse
6415 \DeclareOption{multiple}{\multipletrue}
6416 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6417 \ProcessOptions
```

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

```
6418 \RequirePackage{keyval}[1997/11/10]
6419 \RequirePackage{problem}
```

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

```
6420 \newcommand\hwexam@assignment@kw{Assignment}
6421 \newcommand\hwexam@given@kw{Given}
6422 \newcommand\hwexam@due@kw{Due}
6423 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6424 blank~for~extra~space}
6425 \def\hwexam@minutes@kw{minutes}
6426 \newcommand\correction@probs@kw{prob.}
6427 \newcommand\correction@pts@kw{total}
6428 \newcommand\correction@reached@kw{reached}
6429 \newcommand\correction@sum@kw{Sum}
6430 \newcommand\correction@grade@kw{grade}
6431 \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

```

6432 \AddToHook{begindocument}{
6433 \ltx@ifpackageloaded{babel}{
6434 \makeatletter
6435 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6436 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6437 \input{hwexam-ngerman.ldf}
6438 }
6439 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6440 \input{hwexam-finnish.ldf}
6441 }
6442 \clist_if_in:NnT \l_tmpa_clist {french}{
6443 \input{hwexam-french.ldf}
6444 }
6445 \clist_if_in:NnT \l_tmpa_clist {russian}{
6446 \input{hwexam-russian.ldf}
6447 }
6448 \makeatother
6449 }{}
6450 }
6451

```

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.

```

6452 \newcounter{assignment}
6453 \numberproblemsin{assignment}
6454 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6455 \keys_define:nn { hwexam / assignment } {
6456 id .str_set:N = \l__hwexam_assign_id_str,
6457 number .int_set:N = \l__hwexam_assign_number_int,
6458 title .tl_set:N = \l__hwexam_assign_title_tl,
6459 type .tl_set:N = \l__hwexam_assign_type_tl,
6460 given .tl_set:N = \l__hwexam_assign_given_tl,
6461 due .tl_set:N = \l__hwexam_assign_due_tl,
6462 loadmodules .code:n = {
6463 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6464 }
6465 }
6466 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6467 \str_clear:N \l__hwexam_assign_id_str
6468 \int_set:Nn \l__hwexam_assign_number_int {-1}
6469 \tl_clear:N \l__hwexam_assign_title_tl
6470 \tl_clear:N \l__hwexam_assign_type_tl
6471 \tl_clear:N \l__hwexam_assign_given_tl
6472 \tl_clear:N \l__hwexam_assign_due_tl
6473 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6474 \keys_set:nn { hwexam / assignment }{ #1 }
6475 }

```

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.

```

6476 \newcommand\given@due[2]{
6477 \bool_lazy_all:nF {
6478 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6479 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6480 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6481 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6482 }{ #1 }
6483
6484 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6485 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6486 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6487 }
6488 }{
6489 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6490 }
6491
6492 \bool_lazy_or:nnF {
6493 \bool_lazy_and_p:nn {
6494 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6495 }{
6496 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6497 }
6498 }{
6499 \bool_lazy_and_p:nn {
6500 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6501 }{
6502 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6503 }
6504 }{ ,~ }
6505
6506 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6507 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6508 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6509 }
6510 }{
6511 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6512 }
6513
6514 \bool_lazy_all:nF {
6515 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6516 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6517 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6518 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6519 }{ #2 }
6520 }

```

`\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.

```

6521 \newcommand\assignment@title[3]{
6522 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6523 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6524 #1
6525 }{
6526 #2\l__hwexam_assign_title_tl#3
6527 }
6528 }{
6529 #2\l__hwexam_inclasssign_title_tl#3
6530 }
6531 }

```

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

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

```

6532 \newcommand\assignment@number{
6533 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6534 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6535 \arabic{assignment}
6536 } {
6537 \int_use:N \l__hwexam_assign_number_int
6538 }
6539 }{
6540 \int_use:N \l__hwexam_inclasssign_number_int
6541 }
6542 }

```

(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.

```

6543 \newenvironment{assignment}[1][ ]{
6544 \__hwexam_assignment_args:n { #1 }
6545 %\sref@target
6546 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6547 \global\stepcounter{assignment}
6548 }{
6549 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6550 }
6551 \setcounter{problem}{0}
6552 \def\current@section@level{\document@hwexamtype}
6553 %\sref@label@id{\document@hwexamtype \thesection}
6554 \begin{@assignment}
6555 }{
6556 \end{@assignment}
6557 }

```

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

```

6558 \def\ass@title{
6559 \protect\document@hwexamtype~\arabic{assignment}
6560 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6561 }
6562 \ifmultiple
6563 \newenvironment{@assignment}{
6564 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6565 \begin{omgroup}[loadmodules]{\ass@title}
6566 }{
6567 \begin{omgroup}{\ass@title}
6568 }
6569 }{
6570 \end{omgroup}
6571 }

```

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

```

6572 \else
6573 \newenvironment{@assignment}{
6574 \begin{center}\bf
6575 \Large@title\strut\\
6576 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}
6577 \large\given@due{--;\}\{;\}
6578 \end{center}
6579 }{}
6580 \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.

```

6581 \keys_define:nn { hwexam / inclassignment } {
6582 %id .str_set_x:N = \l__hwexam_assign_id_str,
6583 number .int_set:N = \l__hwexam_inclassign_number_int,
6584 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6585 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6586 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6587 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6588 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6589 }
6590 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6591 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6592 \tl_clear:N \l__hwexam_inclassign_title_tl
6593 \tl_clear:N \l__hwexam_inclassign_type_tl
6594 \tl_clear:N \l__hwexam_inclassign_given_tl
6595 \tl_clear:N \l__hwexam_inclassign_due_tl
6596 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6597 \keys_set:nn { hwexam / inclassignment }{ #1 }
6598 }
6599 \__hwexam_inclassignment_args:n {}
6600
6601 \newcommand\inputassignment[2][{}]{

```

```

6602 \_hwexam_inclassnment_args:n { #1 }
6603 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6604 \input{#2}
6605 }{
6606 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6607 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6608 }
6609 }
6610 \_hwexam_inclassnment_args:n {}
6611 }
6612 \newcommand\includeassignment[2][ ]{
6613 \newpage
6614 \inputassignment[#1]{#2}
6615 }

```

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

42.4 Typesetting Exams

\quizheading

```

6616 \ExplSyntaxOff
6617 \newcommand\quizheading[1]{%
6618 \def\@tas{#1}%
6619 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6620 \ifx\@tas\@empty\else%
6621 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6622 \fi%
6623 }
6624 \ExplSyntaxOn

```

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

\testheading

```

6625
6626 \def\hwexamheader{\input{hwexam-default.header}}
6627
6628 \def\hwexamminutes{
6629 \tl_if_empty:NTF \testheading@duration {
6630 {\testheading@min}~\hwexam@minutes@kw
6631 }{
6632 \testheading@duration
6633 }
6634 }
6635
6636 \keys_define:nn { hwexam / testheading } {
6637 min .tl_set:N = \testheading@min,
6638 duration .tl_set:N = \testheading@duration,
6639 reqpts .tl_set:N = \testheading@reqpts,
6640 tools .tl_set:N = \testheading@tools
6641 }
6642 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6643 \tl_clear:N \testheading@min
6644 \tl_clear:N \testheading@duration

```



```

6645 \tl_clear:N \testheading@reqpts
6646 \tl_clear:N \testheading@tools
6647 \keys_set:nn { hwexam / testheading }{ #1 }
6648 }
6649 \newenvironment{testheading}[1][]{
6650   \_hwexam_testheading_args:n{ #1 }
6651   \newcount\check@time\check@time=\testheading@min
6652   \advance\check@time by -\theassignment@totalmin
6653   \newif\if@bonuspoints
6654   \tl_if_empty:NTF \testheading@reqpts {
6655     \@bonuspointsfalse
6656   }{
6657     \newcount\bonus@pts
6658     \bonus@pts=\theassignment@totalpts
6659     \advance\bonus@pts by -\testheading@reqpts
6660     \edef\bonus@pts{\the\bonus@pts}
6661     \@bonuspointstrue
6662   }
6663   \edef\check@time{\the\check@time}
6664
6665   \makeatletter\hwexamheader\makeatother
6666 }{
6667   \newpage
6668 }

```

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

\testspace

```

6669 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6670 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6671 \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.

```

6672 <@=problems>
6673 \renewcommand\@problem[3]{
6674   \stepcounter{assignment@probs}
6675   \def\__problemspts{#2}
6676   \ifx\__problemspts\@empty\else
6677     \addtocounter{assignment@totalpts}{#2}
6678   \fi
6679   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6680   \xdef\correction@probs{\correction@probs & #1}%
6681   \xdef\correction@pts{\correction@pts & #2}
6682   \xdef\correction@reached{\correction@reached &}

```

```

6683 }
6684 <@@=hwexam>

```

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

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

```

6685 \newcounter{assignment@probs}
6686 \newcounter{assignment@totalpts}
6687 \newcounter{assignment@totalmin}
6688 \def\correction@probs{\correction@probs@kw}
6689 \def\correction@pts{\correction@pts@kw}
6690 \def\correction@reached{\correction@reached@kw}
6691 \stepcounter{assignment@probs}
6692 \newcommand\correction@table{
6693 \resizebox{\textwidth}{!}{%
6694 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6695 &\multicolumn{\theassignment@probs}{c|}||%|
6696 {\footnotesize\correction@forgrading@kw} &\\ \hline
6697 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6698 \correction@pts & \theassignment@totalpts & \\ \hline
6699 \correction@reached & & \[.7cm]\hline
6700 \end{tabular}}
6701 </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}

```