

The sTeX3 Package *

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

2022-03-03

Abstract

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.0 (last revised 2022-03-03)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
3	Using sTeX	6
4	sTeX Archives	7
4.1	The Local MathHub-Directory	7
4.2	The Structure of sTeX Archives	7
4.3	MANIFEST.MF-Files	8
5	Creating New Modules and Symbols	9
5.1	Advanced Structuring Mechanisms	9
5.2	Primitive Symbols (The sTeX Metatheory)	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	34
14.1	Macros and Environments	34
15	<code>sTeX</code>-Terms	36
15.1	Macros and Environments	36
16	<code>sTeX</code>-Structural Features	38
16.1	Macros and Environments	38
16.1.1	Structures	38
17	<code>sTeX</code>-Statements	39
17.1	Macros and Environments	39
18	<code>sTeX</code>-Proofs: Structural Markup for Proofs	40
18.1	Introduction	42
18.2	The User Interface	43
18.2.1	Package Options	43
18.2.2	Proofs and Proof steps	43
18.2.3	Justifications	43
18.2.4	Proof Structure	45
18.2.5	Proof End Markers	45
18.2.6	Configuration of the Presentation	45
18.3	Limitations	46
19	<code>sTeX</code>-Metatheory	47
19.1	Symbols	47
III	Extensions	48

20	Tikzinput	49
20.1	Macros and Environments	49
21	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	50
21.1	Introduction	50
21.2	The User Interface	51
21.2.1	Package and Class Options	51
21.2.2	Document Structure	51
21.2.3	Ignoring Inputs	53
21.2.4	Structure Sharing	53
21.2.5	Global Variables	53
21.2.6	Colors	54
21.3	Limitations	54
22	NotesSlides – Slides and Course Notes	55
22.1	Introduction	55
22.2	The User Interface	55
22.2.1	Package Options	55
22.2.2	Notes and Slides	56
22.2.3	Header and Footer Lines of the Slides	57
22.2.4	Frame Images	57
22.2.5	Colors and Highlighting	58
22.2.6	Front Matter, Titles, etc.	58
22.2.7	Excursions	58
22.2.8	Miscellaneous	59
22.3	Limitations	59
23	problem.sty: An Infrastructure for formatting Problems	60
23.1	Introduction	60
23.2	The User Interface	60
23.2.1	Package Options	60
23.2.2	Problems and Solutions	61
23.2.3	Multiple Choice Blocks	62
23.2.4	Including Problems	62
23.2.5	Reporting Metadata	62
23.3	Limitations	62
24	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	64
24.1	Introduction	65
24.2	The User Interface	65
24.2.1	Package and Class Options	65
24.2.2	Assignments	65
24.2.3	Typesetting Exams	65
24.2.4	Including Assignments	66
24.3	Limitations	66
IV	Implementation	68

25	STeX-Basics Implementation	69
25.1	The STeXDocument Class	69
25.2	Preliminaries	69
25.3	Messages and logging	70
25.4	HTML Annotations	71
25.5	Babel Languages	74
25.6	Auxiliary Methods	75
26	STeX-MathHub Implementation	76
26.1	Generic Path Handling	76
26.2	PWD and kpsewhich	78
26.3	File Hooks and Tracking	79
26.4	MathHub Repositories	80
26.5	Using Content in Archives	84
27	STeX-References Implementation	89
27.1	Document URIs and URLs	89
27.2	Setting Reference Targets	91
27.3	Using References	93
28	STeX-Modules Implementation	96
28.1	The smodule environment	100
28.2	Invoking modules	105
29	STeX-Module Inheritance Implementation	107
29.1	SMS Mode	107
29.2	Inheritance	110
30	STeX-Symbols Implementation	115
30.1	Symbol Declarations	115
30.2	Notations	121
30.3	Variables	131
31	STeX-Terms Implementation	135
31.1	Symbol Invocations	135
31.2	Terms	141
31.3	Notation Components	145
31.4	Variables	147
32	STeX-Structural Features Implementation	150
32.1	Imports with modification	151
32.2	The feature environment	157
32.3	Structure	158
33	STeX-Statements Implementation	166
33.1	Definitions	166
33.2	Assertions	171
33.3	Examples	174
33.4	Logical Paragraphs	177

34 The Implementation	182
34.1 Package Options	182
34.2 Proofs	182
34.3 Justifications	193
35 \TeX-Others Implementation	195
36 \TeX-Metatheory Implementation	196
37 Tikzinput Implementation	199
38 document-structure.sty Implementation	201
38.1 The document-structure Class	201
38.2 Class Options	201
38.3 Beefing up the <code>document</code> environment	202
38.4 Implementation: document-structure Package	202
38.5 Package Options	202
38.6 Document Structure	204
38.7 Front and Backmatter	207
38.8 Global Variables	209
39 NotesSlides – Implementation	210
39.1 Class and Package Options	210
39.2 Notes and Slides	212
39.3 Header and Footer Lines	216
39.4 Frame Images	217
39.5 Colors and Highlighting	218
39.6 Sectioning	219
39.7 Excursions	222
40 The Implementation	223
40.1 Package Options	223
40.2 Problems and Solutions	224
40.3 Multiple Choice Blocks	230
40.4 Including Problems	231
40.5 Reporting Metadata	232
41 Implementation: The hwexam Class	234
41.1 Class Options	234
42 Implementation: The hwexam Package	236
42.1 Package Options	236
42.2 Assignments	237
42.3 Including Assignments	240
42.4 Typesetting Exams	241
42.5 Leftovers	243

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,
- RuS_{TeX} to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

Having set everything up, we can write a first s_TE_X document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder.

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

s_TE_X now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitiesum` later on.

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using \LaTeX

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

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

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

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

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

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

TODO: [terms documentation](#)

TODO: [references documentation](#)

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

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

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

```
\begin{smodule}{assoctest}
\syndef{foo}[args=1a]{\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: $a:w_1;b:w_2;c:[w_1;x+[w_1;y+z;w_2];w_2]$

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

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

```
\begin{smodule}{magma}
\syndef{universe}{\comp{\mathcal U}}
\syndef{operation}[args=2,op=\circ]{#1\comp{\circ}#2}
\end{smodule}
\begin{smodule}{monoid}
\importmodule{magma}
\syndef{unit}{\comp{e}}
\end{smodule}
\begin{smodule}{group}
\importmodule{monoid}
\syndef{inverse}[args=1]{#1^{\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}{ruminus}
\end{copymodule}
\notation*{rplus}[plus,op=+,prec=60]{#1 \comp+ #2}
\notation*{rzero}{zero}{\comp0}
\notation*{ruminus}{uminus,op=-}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedcl[name=times]{operation}{rtimes}
\renamedcl[name=one]{unit}{rone}
\end{copymodule}
\notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
\notation*{rone}{one}{\comp1}
Test:  $\$ \rtimes a \{ \plus c \{ \rtimes de \} \$$ 
\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{plus}[args=2,op=+]{#1 \comp+ #2}
\symdef{zero}{\comp0}
\symdef{uminus}[args=1,op=-]{\comp-#1}

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

Module 6:

5.2 Primitive Symbols (The $\text{\texttt{STEX}}$ Metatheory)

TODO: metatheory documentation

Chapter 6

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

TODO: statements documentation
TODO: sproofs documentation

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

Chapter 8

Stuff

8.1 Modules

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

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 5

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

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

Example 6

```
\notation{mult}[cdot]{#1 \comp{\cdot} #2}
\notation{mult}[times]{#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

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

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

.

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

Example 8

```
\mult{\comp{Multiplying} \arg*\mathmult{a}{b}} again by \arg{b} yields ...
```

Multiplying again by b yields...

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

Example 9

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

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

.

⁴EdNOTE: TODO

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

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

Example 10

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

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

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

Example 11

```
\mult!{\comp{Multiplication}} (denoted by  $\textcolor{blue}{\cdot}$ ) is defined by...
```

$\textcolor{blue}{\cdot}$ (denoted by $\textcolor{blue}{\cdot}$) is defined by...

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

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

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

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

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

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two *i*-type arguments.

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

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

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

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

Example 12

```
\symdef{mult}[args=a]{#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{numseq}[args=ai]{#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 _x \int _y \int _z f dx dy dz`?

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

Precedences

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

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

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

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

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

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

For example:

Module 9:

Example 14

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

$a+b\cdot c$ 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{<i>archive-ID</i>}{<i>filename</i>}</code>
	Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\mhinput</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (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[<i>archive-ID</i>]{<i>filename</i>}</code> Adds a <code>.bib</code> -file <i>filename</i> in archive <i>archive-ID</i> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<code>\libinput{<i>filename</i>}</code> Inputs <i>filename.tex</i> 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[<i>args</i>]{<i>filename</i>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<i>meta</i>{<i>args</i>}]<i>Arg</i>{<i>filename</i>}</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> <hr/> <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> <hr/> <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

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

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

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

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

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

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

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

Chapter 12

STEX-Modules

This sub package contains code related to Modules

12.1 Macros and Environments

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

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

12.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

`\stex_module_setup:nn` `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`

Sets up a new module with name `\langle name \rangle` and optional parameters `\langle params \rangle`. In particular, sets `\l_stex_current_module_str` appropriately.

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

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

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

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

`\stex_invoke_module:n`

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

`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

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

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

Tests whether SMS mode is currently active.

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

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

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

13.1.2 Imports and Inheritance

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

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

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

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

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

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

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

1. If `<archive-ID>` is empty:
 - (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.
 - (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.
2. Otherwise:
 - (a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`. That module should lie directly in the namespace of the archive.
 - (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

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

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

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

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

Chapter 14

STEX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

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

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

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

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

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

<hr/> <hr/>	
<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	
<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code>
	Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \SIX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	
<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
	Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \SIX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <hr/>	
<code>\stex_term_custom:nn</code>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code>
	Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <hr/>	
<code>\stex_highlight_term:nn</code>	<code>\stex_highlight_term:nn{⟨URI⟩}{⟨args⟩}</code>
	Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.
<hr/> <hr/>	
<code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@uri</code>	<code>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <hr/>	
<code>\STEXinvisible</code>	Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <hr/>	
<code>\ellipses</code>	TODO

Chapter 16

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

18.2.4 Proof Structure

subproof The pfcases environment is used to mark up a subproof. This environment takes an optional KeyVal argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the proof environment). The method key can be used to give the name of the proof method executed to make this subproof.

method

spfcases The pfcases environment is used to mark up a proof by cases. Technically it is a variant of the subproof where the method is by-cases. Its contents are spfcase environments that mark up the cases one by one.

spfcase The content of a pfcases environment are a sequence of case proofs marked up in the pfcase 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 pfcase environment is the same as that of a proof, i.e. steps, proofcomments, and pfcases environments. \spfcasesketch is a variant of the spfcase environment that takes the same arguments, but instead of the spfsteps in the body uses a third argument for a proof sketch.

\spfcasesketch

sproofcomment The proofcomment environment is much like a step, only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a \premise.

18.2.5 Proof End Markers

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

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

\sProofEndSymbol

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

18.2.6 Configuration of the Presentation

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

Environment	configuration macro	value
sproof	\spf@proof@kw	Proof
sketchproof	\spf@sketchproof@kw	Proof Sketch

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

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

18.3 Limitations

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

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

21.1 Introduction

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

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

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

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

21.2 The User Interface

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

21.2.1 Package and Class Options

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

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

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

21.2.2 Document Structure

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

```

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

```

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

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

`blindfragment`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

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

22.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

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

22.2.3 Header and Footer Lines of the Slides

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

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

`\setsource`
`\setlicensing`

22.2.4 Frame Images

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

`\frameimage`
`\mhframeimage`

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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


22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

`assignment` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional `KeyVal` argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — 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 \LaTeX source.

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

25.3 Messages and logging

```

41 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

25.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`
`\latexml_if:TF`

```

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

```

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

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

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

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

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

```

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

```

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

```

```

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

```

```

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

```

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

25.5 Babel Languages

```

213 <@=stex_language>

```

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

```

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

```



```

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

```

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

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

```

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

```

25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\ignorespacesandpars`

```

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

```

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

Chapter 26

STEX -MathHub Implementation

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

26.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```

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

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

26.3 File Hooks and Tracking

```

387 <@@=stex_files>

```

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

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

`\g__stex_files_stack`

keeps track of file changes

```

388 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

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

`\g_stex_currentfile_seq`

```

392 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

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

```

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

`\stex_filestack_pop:`

```

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

```

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

Hooks for the current file:

```

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

```

26.4 MathHub Repositories

420 $\langle @@=\text{stex_mathhub} \rangle$

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

\c_stex_mathhub_str

```

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

```

```

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

```

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

```

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

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

464 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

```

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

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```
500 \ior_new:N \c__stex_mathhub_manifest_ior
```

(End definition for \c__stex_mathhub_manifest_ior.)

__stex_mathhub_parse_manifest:n Stores the entries in manifest file in the corresponding property list:

```

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

```



```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

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

```

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

```

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

```

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

26.5 Using Content in Archives

`\mhpath`

```

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

```

```

615 }
616 }

```

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

`\inputref`
`\mhinput`

```

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

```

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

`\addmhbibresource`

```

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

```

```

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

```

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

\libinput

```

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

```

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

\libusepackage

```

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

```

```

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

```

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

`\mhgraphics`
`\cmhgraphics`

```

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

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```

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

Chapter 27

STEX -References Implementation

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

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

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

`\STEXreftitle`

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

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

27.1 Document URIs and URLs

`\l_stex_current_docns_str`

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

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

`\stex_get_document_uri:`

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

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

`\l_stex_current_docurl_str`

```
814 \str_new:N \l_stex_current_docurl_str
```

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

`\stex_get_document_url:`

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



```

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

```

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

27.2 Setting Reference Targets

```

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

```

`\stex_ref_new_doc_target:n`

```

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

```

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

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

```

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

```

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

```

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

```

`\stex_ref_new_sym_target:n`

```

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

```

```

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

```

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

27.3 Using References

```

926 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

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

```

The actual macro:

```

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

```

```

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

```

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

`\srefsym`

```

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

```

```

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

```

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

\srefsymuri

```

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

```

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

```

1052 </package>

```

Chapter 28

STEX -Modules Implementation

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

`\l_stex_current_module_str` The current module:

```
1075 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 28.)

`\l_stex_all_modules_seq` Stores all available modules

```
1076 \seq_new:N \l_stex_all_modules_seq
```

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

```

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

```

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

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

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

```

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

```

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

```

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

```

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

```

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

`\stex_collect_imports:n`

```

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

```

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

`\stex_do_up_to_module:n`

```

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

```

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

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

1146

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

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

```

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

```



```

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

```

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

28.1 The smodule environment

smodule arguments:

```

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

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

```

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

```

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

```

Next, we determine the language of the module:

```

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

```

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

```

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

```

We load the metatheory:

```

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

```

```

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

```

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

smodule The module environment.

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

```

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

```

(End definition for `_stex_modules_begin_module:.`)

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

```

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

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1390 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1391 \NewDocumentEnvironment { smodule } { 0{} m } {
1392   \stex_module_setup:nn{#1}{#2}
1393   \par
1394   \stex_if_smsmode:F{
1395     \tl_clear:N \l_tmpa_tl
1396     \clist_map_inline:Nn \smoduletype {
1397       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1398         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1399       }
1400     }
1401     \tl_if_empty:NTF \l_tmpa_tl {
1402       \__stex_modules_smodule_start:
1403     }{
1404       \l_tmpa_tl
1405     }
1406   }
1407   \__stex_modules_begin_module:
1408   \str_if_empty:NF \smoduleid {
1409     \stex_ref_new_doc_target:n \smoduleid
1410   }
1411   \stex_smsmode_do:
1412 } {
1413   \__stex_modules_end_module:
1414   \stex_if_smsmode:F {
1415     \end{stex_annotate_env}
1416     \clist_set:No \l_tmpa_clist \smoduletype
1417     \tl_clear:N \l_tmpa_tl
1418     \clist_map_inline:Nn \l_tmpa_clist {
1419       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1420         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1421       }
1422     }
1423     \tl_if_empty:NTF \l_tmpa_tl {
1424       \__stex_modules_smodule_end:
1425     }{
1426       \l_tmpa_tl
1427     }
1428   }
1429 }

```

`\stexpatchmodule`

```

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

```

```

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

```

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

28.2 Invoking modules

```

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

```

```
1486 }
```

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

`\stex_activate_module:n`

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

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

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


Chapter 29

STEX -Module Inheritance Implementation

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

29.1 SMS Mode

```
1504 <@@=stex_smsmode>

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

```

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

```

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

`\stex_if_smsmode_p:`
`\stex_if_smsmode:TF`

```

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

```

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

`_stex_smsmode_in_smsmode:nn`

```

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

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

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

```

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

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

```

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

```

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

29.2 Inheritance

```

1619 <@@=stex_importmodule>

\stex_import_module_uri:nn

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

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

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

1663 \str_new:N \l_stex_import_ns_str

(End definition for \l_stex_import_name_str and others. These variables are documented on page 33.)

```

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

```

```

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

```

```

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

```

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

`\importmodule`

```

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

```

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

`\usemodule`

```

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

```

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

1813 `\endpackage`

Chapter 30

STEX -Symbols Implementation

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

30.1 Symbol Declarations

```
1825 <@@=stex_symdecl>

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

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

\STEXsymbol
1834 \NewDocumentCommand \STEXsymbol { m } {
1835   \stex_get_symbol:n { #1 }
1836   \exp_args:No
1837   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1838 }
```

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

`symdecl` arguments:

```

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

```

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

```

1867
1868 \NewDocumentCommand \symdecl { s m O{} } {
1869   \__stex_symdecl_args:n { #3 }
1870   \IfBooleanTF #1 {
1871     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1872   } {
1873     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1874   }
1875   \stex_symdecl_do:n { #2 }
1876   \stex_smsmode_do:
1877 }
1878
1879 \cs_new_protected:Nn \stex_symdecl_do:nn {
1880   \__stex_symdecl_args:n{#1}
1881   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1882   \stex_symdecl_do:n{#2}
1883 }
1884
1885 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

`\stex_symdecl_do:n`

```
1886 \cs_new_protected:Nn \stex_symdecl_do:n {
1887   \stex_if_in_module:F {
1888     % TODO throw error? some default namespace?
1889   }
1890
1891   \str_if_empty:NT \l_stex_symdecl_name_str {
1892     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1893   }
1894
1895   \prop_if_exist:cT { l_stex_symdecl_
1896     \l_stex_current_module_str ?
1897     \l_stex_symdecl_name_str
1898   }_prop
1899   }{
1900     % TODO throw error (beware of circular dependencies)
1901   }
1902
1903   \prop_clear:N \l_tmpa_prop
1904   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1905   \seq_clear:N \l_tmpa_seq
1906   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1907   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1908
1909   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1910     \str_if_empty:NF \l_stex_module_deprecate_str {
1911       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1912     }
1913   }
1914   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1915
1916   \exp_args:No \stex_add_constant_to_current_module:n {
1917     \l_stex_symdecl_name_str
1918   }
1919
1920   % arity/args
1921   \int_zero:N \l_tmpb_int
1922
1923   \bool_set_true:N \l_tmpa_bool
1924   \str_map_inline:Nn \l_stex_symdecl_args_str {
1925     \token_case_meaning:NnF ##1 {
1926       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1927       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1928       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1929       {\tl_to_str:n a} {
1930         \bool_set_false:N \l_tmpa_bool
1931         \int_incr:N \l_tmpb_int
1932       }
1933       {\tl_to_str:n B} {
1934         \bool_set_false:N \l_tmpa_bool
1935         \int_incr:N \l_tmpb_int
1936       }
1937     }{
1938       \msg_error:nnxx{stex}{error/wrongargs}{
```

```

1939         \l_stex_current_module_str ?
1940         \l_stex_symdecl_name_str
1941     }{##1}
1942 }
1943 }
1944 \bool_if:NTF \l_tmpa_bool {
1945     % possibly numeric
1946     \str_if_empty:NTF \l_stex_symdecl_args_str {
1947         \prop_put:Nnn \l_tmpa_prop { args } {}
1948         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1949     }{
1950         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1951         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1952         \str_clear:N \l_tmpa_str
1953         \int_step_inline:nn \l_tmpa_int {
1954             \str_put_right:Nn \l_tmpa_str i
1955         }
1956         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1957     }
1958 } {
1959     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1960     \prop_put:Nnx \l_tmpa_prop { arity }
1961     { \str_count:N \l_stex_symdecl_args_str }
1962 }
1963 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1964
1965
1966 % semantic macro
1967
1968 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1969     \exp_args:Nx \stex_do_up_to_module:n {
1970         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1971             \l_stex_current_module_str ? \l_stex_symdecl_name_str
1972         }}
1973     }
1974
1975     \bool_if:NF \l_stex_symdecl_local_bool {
1976         \exp_args:Nx \stex_add_to_current_module:n {
1977             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1978                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979             } }
1980         }
1981     }
1982 }
1983
1984 \stex_debug:nn{symbols}{New~symbol:~
1985     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1986     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1987     Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
1988     Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
1989 }
1990
1991 % circular dependencies require this:
1992

```

```

1993 \prop_if_exist:cF {
1994   l_stex_symdecl_
1995   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1996   _prop
1997 } {
1998   \exp_args:Nx \stex_do_up_to_module:n {
1999     \prop_set_from_keyval:cn {
2000       l_stex_symdecl_
2001       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002       _prop
2003     } {\prop_to_keyval:N \l_tmpa_prop}
2004   }
2005 }
2006
2007 \seq_clear:c {
2008   l_stex_symdecl_
2009   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2010   _notations
2011 }
2012
2013 \bool_if:NF \l_stex_symdecl_local_bool {
2014   \exp_args:Nx
2015   \stex_add_to_current_module:n {
2016     \seq_clear:c {
2017       l_stex_symdecl_
2018       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2019       _notations
2020     }
2021     \prop_set_from_keyval:cn {
2022       l_stex_symdecl_
2023       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2024       _prop
2025     } {
2026       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2027       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2028       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2029       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2030       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2031       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2032     }
2033   }
2034 }
2035
2036 \stex_if_smsmode:F {
2037 %   \exp_args:Nx \stex_do_up_to_module:n {
2038 %     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2039 %       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2040 %     }
2041 %   }
2042 \stex_if_do_html:T {
2043   \stex_annotate_invisible:nnn {symdecl} {
2044     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2045   } {
2046     \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st

```

```

2047 \stex_annotate_invisible:nnn{args}{}{
2048 \prop_item:Nn \l_tmpa_prop { args }
2049 }
2050 \stex_annotate_invisible:nnn{macroname}{#1}{
2051 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2052 \stex_annotate_invisible:nnn{definiens}{
2053 { $\l_stex_symdecl_definiens_tl$ }
2054 }
2055 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2056 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{
2057 }
2058 }
2059 }
2060 }
2061 }

```

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

`\stex_get_symbol:n`

```

2062 \str_new:N \l_stex_get_symbol_uri_str
2063
2064 \cs_new_protected:Nn \stex_get_symbol:n {
2065 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2066 \tl_set:Nn \l_tmpa_tl { #1 }
2067 \__stex_symdecl_get_symbol_from_cs:
2068 }{
2069 % argument is a string
2070 % is it a command name?
2071 \cs_if_exist:cTF { #1 }{
2072 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2073 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2074 \str_if_empty:NNTF \l_tmpa_str {
2075 \exp_args:Nx \cs_if_eq:NNTF {
2076 \tl_head:N \l_tmpa_tl
2077 } \stex_invoke_symbol:n {
2078 \__stex_symdecl_get_symbol_from_cs:
2079 }{
2080 \__stex_symdecl_get_symbol_from_string:n { #1 }
2081 }
2082 } {
2083 \__stex_symdecl_get_symbol_from_string:n { #1 }
2084 }
2085 }{
2086 % argument is not a command name
2087 \__stex_symdecl_get_symbol_from_string:n { #1 }
2088 % \l_stex_all_symbols_seq
2089 }
2090 }
2091 \str_if_eq:eeF {
2092 \prop_item:cn {
2093 \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2094 }{ deprecate }
2095 }{}{
2096 \msg_warning:nxxx{stex}{warning/deprecated}{

```

```

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

```

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

30.2 Notations

```

2140 <@@=stex_notation>

      notation arguments:
2141 \keys_define:nn { stex / notation } {
2142   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2143   variant   .tl_set_x:N = \l__stex_notation_variant_str ,

```

```

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

```

\notation

```

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

```

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

\stex_notation_do:nnnnn

```

2179 \seq_new:N \l__stex_notation_precedences_seq
2180 \tl_new:N \l__stex_notation_opprec_tl
2181 \int_new:N \l__stex_notation_currarg_int
2182 \tl_new:N \stex_symbol_after_invokation_tl
2183
2184 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2185 \let\l_stex_current_symbol_str\relax
2186 \seq_clear:N \l__stex_notation_precedences_seq
2187 \tl_clear:N \l__stex_notation_opprec_tl
2188 \str_set:Nx \l__stex_notation_args_str { #1 }
2189 \str_set:Nx \l__stex_notation_arity_str { #2 }
2190 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2191 \str_set:Nx \l__stex_notation_prec_str { #4 }
2192

```



```

2193 % precedences
2194 \str_if_empty:NTF \l__stex_notation_prec_str {
2195   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2196     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2197   }{
2198     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2199   }
2200 } {
2201   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2202     \tl_set:No \l__stex_notation_opprec_tl { \neginfpref }
2203     \int_step_inline:nn { \l__stex_notation_arity_str } {
2204       \exp_args:NNo
2205       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infpref }
2206     }
2207   }{
2208     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2209     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2210       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2211       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2212         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2213           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2214         \seq_map_inline:Nn \l_tmpa_seq {
2215           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2216         }
2217       }
2218     }{
2219       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2220         \tl_set:No \l__stex_notation_opprec_tl { \infpref }
2221       }{
2222         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2223       }
2224     }
2225   }
2226 }
2227
2228 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2229 \int_step_inline:nn { \l__stex_notation_arity_str } {
2230   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2231     \exp_args:NNo
2232     \seq_put_right:No \l__stex_notation_precedences_seq {
2233       \l__stex_notation_opprec_tl
2234     }
2235   }
2236 }
2237 \tl_clear:N \l_stex_notation_dummyargs_tl
2238
2239 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2240   \exp_args:NNe
2241   \cs_set:Npn \l_stex_notation_macrocode_cs {
2242     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2243     { \l__stex_notation_suffix_str }
2244     { \l__stex_notation_opprec_tl }
2245     { \exp_not:n { #5 } }
2246   }

```

```

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

```

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

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

```

2289 \cs_new_protected:Nn \__stex_notation_arguments: {
2290 \int_incr:N \l__stex_notation_currarg_int
2291 \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2292 \l_stex_notation_after_do_tl
2293 }{
2294 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2295 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2296 \str_if_eq:VnTF \l_tmpa_str a {

```

```

2297     \_stex_notation_argument_assoc:n
2298   }{
2299     \str_if_eq:VnTF \l_tmpa_str B {
2300       \_stex_notation_argument_assoc:n
2301     }{
2302       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2303       \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2304         { \_stex_term_math_arg:nnn
2305           { \int_use:N \l__stex_notation_currarg_int }
2306           { \l_tmpa_str }
2307           { #####\int_use:N \l__stex_notation_currarg_int }
2308         }
2309       }
2310       \_stex_notation_arguments:
2311     }
2312   }
2313 }
2314 }

```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2315 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2316
2317   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2318     {\l__stex_notation_arity_str}{
2319       #1
2320     }
2321   \int_zero:N \l_tmpa_int
2322   \tl_clear:N \l_tmpa_tl
2323   \str_map_inline:Nn \l__stex_notation_args_str {
2324     \int_incr:N \l_tmpa_int
2325     \tl_put_right:Nx \l_tmpa_tl {
2326       \str_if_eq:nnTF {##1}{a}{ {} }{
2327         \str_if_eq:nnTF {##1}{B}{ {} }{
2328           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
2329         }
2330       }
2331     }
2332   }
2333   \exp_after:wN\exp_after:wN\exp_after:wN \def
2334   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2335   \exp_after:wN\exp_after:wN\exp_after:wN ##
2336   \exp_after:wN\exp_after:wN\exp_after:wN 1
2337   \exp_after:wN\exp_after:wN\exp_after:wN ##
2338   \exp_after:wN\exp_after:wN\exp_after:wN 2
2339   \exp_after:wN\exp_after:wN\exp_after:wN {
2340     \exp_after:wN \exp_after:wN \exp_after:wN
2341     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2342       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2343     }
2344   }
2345
2346   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str

```

```

2347 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2348   \stex_term_math_assoc_arg:nnnn
2349   { \int_use:N \l__stex_notation_currarg_int }
2350   { \l_tmpa_str }
2351   { ####\int_use:N \l__stex_notation_currarg_int }
2352   { \l_tmpa_cs {####1} {####2} }
2353 } }
2354 \__stex_notation_arguments:
2355 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2356 \cs_new_protected:Nn \__stex_notation_final: {
2357   \exp_args:Nne \use:nn
2358   {
2359     \cs_generate_from_arg_count:cNnn {
2360       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2361       \l__stex_notation_suffix_str
2362       _cs
2363     }
2364     \cs_set:Npn \l__stex_notation_arity_str { { {
2365       \exp_after:wN \exp_after:wN \exp_after:wN
2366       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2367       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2368     } } }
2369
2370     \tl_if_empty:NF \l__stex_notation_op_tl {
2371       \cs_set:cpx {
2372         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2373         \l__stex_notation_suffix_str
2374         _cs
2375       } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2376     }
2377
2378     \exp_args:Ne
2379     \stex_add_to_current_module:n {
2380       \cs_generate_from_arg_count:cNnn {
2381         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2382         \l__stex_notation_suffix_str
2383         _cs
2384       } \cs_set:Npn {\l__stex_notation_arity_str} {
2385         \exp_after:wN \exp_after:wN \exp_after:wN
2386         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2387         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2388       }
2389       \tl_if_empty:NF \l__stex_notation_op_tl {
2390         \cs_set:cpn {
2391           stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2392           \l__stex_notation_suffix_str
2393           _cs
2394         } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2395       }
2396     }

```

```

2397 %\exp_args:Nx
2398 % \stex_do_up_to_module:n {
2399   \seq_put_right:cx {
2400     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2401     _notations
2402   } {
2403     \l__stex_notation_suffix_str
2404   }
2405 % }
2406
2407 \stex_debug:nn{symbols}{
2408   Notation~\l__stex_notation_suffix_str
2409   ~for~\l_stex_get_symbol_uri_str^^J
2410   Operator~precedence::~\l__stex_notation_opprec_tl^^J
2411   Argument~precedences::~
2412   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2413   Notation: \cs_meaning:c {
2414     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2415     \l__stex_notation_suffix_str
2416     _cs
2417   }
2418 }
2419
2420 \exp_args:Ne
2421 \stex_add_to_current_module:n {
2422   \seq_put_right:cn {
2423     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2424     _notations
2425   } { \l__stex_notation_suffix_str }
2426 }
2427
2428 \stex_if_smsmode:F {
2429
2430   % HTML annotations
2431   \stex_if_do_html:T {
2432     \stex_annotate_invisible:nnn { notation }
2433     { \l_stex_get_symbol_uri_str } {
2434       \stex_annotate_invisible:nnn { notationfragment }
2435       { \l__stex_notation_suffix_str }{}
2436     }
2437     \stex_annotate_invisible:nnn { precedence }
2438     { \l__stex_notation_prec_str }{}
2439
2440     \int_zero:N \l_tmpa_int
2441     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2442     \tl_clear:N \l_tmpa_tl
2443     \int_step_inline:nn { \l__stex_notation_arity_str }{
2444       \int_incr:N \l_tmpa_int
2445       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2446       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2447       \str_if_eq:VnTF \l_tmpb_str a {
2448         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2449           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2450           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2451         } }

```

```

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

```

(End definition for _stex_notation_final:.)

\setnotation

```

2475 \keys_define:nn { stex / setnotation } {
2476     lang .tl_set_x:N = \l__stex_notation_lang_str ,
2477     variant .tl_set_x:N = \l__stex_notation_variant_str ,
2478     unknown .code:n = \str_set:Nx
2479         \l__stex_notation_variant_str \l_keys_key_str
2480 }
2481
2482 \cs_new_protected:Nn \stex_setnotation_args:n {
2483     \str_clear:N \l__stex_notation_lang_str
2484     \str_clear:N \l__stex_notation_variant_str
2485     \keys_set:nn { stex / setnotation } { #1 }
2486 }
2487
2488 \cs_new_protected:Nn \stex_setnotation:n {
2489     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2490     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2491         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2492         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2493         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2494         { \c_hash_str }
2495         \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2496         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2497     \exp_args:Nx \stex_add_to_current_module:n {
2498         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2499         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2500     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }

```

```

2501         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2502         \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2503         { \c_hash_str }
2504     }
2505     \stex_debug:nn {notations}{
2506         Setting~default~notation~
2507         {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2508         #1 \\
2509         \expandafter\meaning\csname
2510         l_stex_symdecl_#1 _notations\endcsname
2511     }
2512     ){
2513         % todo throw error
2514     }
2515 }
2516
2517 \NewDocumentCommand \setnotation {m m} {
2518     \stex_get_symbol:n { #1 }
2519     \_stex_setnotation_args:n { #2 }
2520     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2521     \stex_smsmode_do:
2522 }
2523
2524 \cs_new_protected:Nn \stex_copy_notations:nn {
2525     \stex_debug:nn {notations}{
2526         Copying~notations~from~#2~to~#1\\
2527         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2528     }
2529     \tl_clear:N \l_tmpa_tl
2530     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2531         \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2532     }
2533     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2534         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2535         \edef \l_tmpa_tl {
2536             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2537             \exp_after:wN\exp_after:wN\exp_after:wN {
2538                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2539             }
2540         }
2541         \exp_args:Nx
2542         \stex_do_up_to_module:n {
2543             \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2544             \cs_generate_from_arg_count:cNnn {
2545                 stex_notation_ #1 \c_hash_str ##1 _cs
2546             } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2547                 \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2548             }
2549         }
2550     }
2551 }
2552
2553 \NewDocumentCommand \copynotation {m m} {
2554     \stex_get_symbol:n { #1 }

```

```

2555 \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2556 \stex_get_symbol:n { #2 }
2557 \exp_args:Noo
2558 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2559 \exp_args:Nx \stex_add_import_to_current_module:n{
2560   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2561 }
2562 \stex_smsmode_do:
2563 }
2564

```

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

\symdef

```

2565 \keys_define:nn { stex / symdef } {
2566   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2567   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2568   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2569   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2570   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2571   op        .tl_set:N = \l__stex_notation_op_tl ,
2572   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2573   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2574   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2575   assoc     .choices:nn =
2576     {bin,binl,binr,pre,conj,pwconj}
2577     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2578   unknown   .code:n = \str_set:Nx
2579     \l__stex_notation_variant_str \l_keys_key_str
2580 }
2581
2582 \cs_new_protected:Nn \l__stex_notation_symdef_args:n {
2583   \str_clear:N \l_stex_symdecl_name_str
2584   \str_clear:N \l_stex_symdecl_args_str
2585   \str_clear:N \l_stex_symdecl_assoctype_str
2586   \bool_set_false:N \l_stex_symdecl_local_bool
2587   \tl_clear:N \l_stex_symdecl_type_tl
2588   \tl_clear:N \l_stex_symdecl_definiens_tl
2589   \str_clear:N \l__stex_notation_lang_str
2590   \str_clear:N \l__stex_notation_variant_str
2591   \str_clear:N \l__stex_notation_prec_str
2592   \tl_clear:N \l__stex_notation_op_tl
2593
2594   \keys_set:nn { stex / symdef } { #1 }
2595 }
2596
2597 \NewDocumentCommand \symdef { m O{} } {
2598   \l__stex_notation_symdef_args:n { #2 }
2599   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2600   \stex_symdecl_do:n { #1 }
2601   \tl_set:Nn \l_stex_notation_after_do_tl {
2602     \l__stex_notation_final:
2603     \stex_smsmode_do:
2604   }

```



```

2605 \str_set:Nx \l_stex_get_symbol_uri_str {
2606   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2607 }
2608 \exp_args:Nx \stex_notation_do:nnnnn
2609 { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2610 { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2611 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2612 { \l__stex_notation_prec_str}
2613 }
2614 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

30.3 Variables

```

2615 <@@=stex_variables>
2616
2617 \keys_define:nn { stex / vardef } {
2618   name      .str_set_x:N = \l__stex_variables_name_str ,
2619   args      .str_set_x:N = \l__stex_variables_args_str ,
2620   type      .tl_set:N    = \l__stex_variables_type_tl ,
2621   def       .tl_set:N    = \l__stex_variables_def_tl ,
2622   op        .tl_set:N    = \l__stex_variables_op_tl ,
2623   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2624   assoc     .choices:nn =
2625     {bin,binl,binr,pre,conj,pwconj}
2626     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2627   bind      .choices:nn =
2628     {forall,exists}
2629     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2630 }
2631
2632 \cs_new_protected:Nn \__stex_variables_args:n {
2633   \str_clear:N \l__stex_variables_name_str
2634   \str_clear:N \l__stex_variables_args_str
2635   \str_clear:N \l__stex_variables_prec_str
2636   \str_clear:N \l__stex_variables_assoctype_str
2637   \str_clear:N \l__stex_variables_bind_str
2638   \tl_clear:N \l__stex_variables_type_tl
2639   \tl_clear:N \l__stex_variables_def_tl
2640   \tl_clear:N \l__stex_variables_op_tl
2641
2642   \keys_set:nn { stex / vardef } { #1 }
2643 }
2644
2645 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2646   \__stex_variables_args:n {#2}
2647   \str_if_empty:NT \l__stex_variables_name_str {
2648     \str_set:Nx \l__stex_variables_name_str { #1 }
2649   }
2650   \prop_clear:N \l_tmpa_prop
2651   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2652
2653   \int_zero:N \l_tmpb_int

```

```

2654 \bool_set_true:N \l_tmpa_bool
2655 \str_map_inline:Nn \l__stex_variables_args_str {
2656   \token_case_meaning:NnF ##1 {
2657     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2658     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2659     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2660     {\tl_to_str:n a} {
2661       \bool_set_false:N \l_tmpa_bool
2662       \int_incr:N \l_tmpb_int
2663     }
2664     {\tl_to_str:n B} {
2665       \bool_set_false:N \l_tmpa_bool
2666       \int_incr:N \l_tmpb_int
2667     }
2668   }{
2669     \msg_error:nnxx{stex}{error/wrongargs}{
2670       variable~\l__stex_variables_name_str
2671     }{##1}
2672   }
2673 }
2674 \bool_if:NTF \l_tmpa_bool {
2675   % possibly numeric
2676   \str_if_empty:NTF \l__stex_variables_args_str {
2677     \prop_put:Nnn \l_tmpa_prop { args } {}
2678     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2679   }{
2680     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2681     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2682     \str_clear:N \l_tmpa_str
2683     \int_step_inline:nn \l_tmpa_int {
2684       \str_put_right:Nn \l_tmpa_str i
2685     }
2686     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2687     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2688   }
2689 } {
2690   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2691   \prop_put:Nnx \l_tmpa_prop { arity }
2692     { \str_count:N \l__stex_variables_args_str }
2693 }
2694 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2695 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2696
2697 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2698
2699 \tl_if_empty:NF \l__stex_variables_op_tl {
2700   \cs_set:cpx {
2701     stex_var_op_notation_\l__stex_variables_name_str_cs
2702   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2703 }
2704
2705 \tl_set:Nn \l_stex_notation_after_do_tl {
2706   \exp_args:Nne \use:nn {
2707     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }

```

```

2708     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2709 } {{
2710   \exp_after:wN \exp_after:wN \exp_after:wN
2711   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2712   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2713 } }
2714 \stex_if_do_html:T {
2715   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2716     \stex_annotate_invisible:nnn { precedence }
2717     { \l__stex_variables_prec_str }{}
2718     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l__stex_variables_type_str }
2719     \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2720     \stex_annotate_invisible:nnn{macroname}{#1}{}
2721     \tl_if_empty:NF \l__stex_variables_def_tl {
2722       \stex_annotate_invisible:nnn{definiens}{}
2723       { $\l__stex_variables_def_tl$ }
2724     }
2725     \str_if_empty:NF \l__stex_variables_assoctype_str {
2726       \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2727     }
2728     \int_zero:N \l_tmpa_int
2729     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2730     \tl_clear:N \l_tmpa_tl
2731     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2732       \int_incr:N \l_tmpa_int
2733       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2734       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2735       \str_if_eq:VnTF \l_tmpb_str a {
2736         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2737           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2738           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2739         } }
2740       }{
2741         \str_if_eq:VnTF \l_tmpb_str B {
2742           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2743             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2744             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2745           } }
2746         }{
2747           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2748             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2749           } }
2750         }
2751       }
2752     }
2753     \stex_annotate_invisible:nnn { notationcomp }{}{
2754       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2755       $ \exp_args:Nno \use:nn { \use:c {
2756         stex_var_notation_\l__stex_variables_name_str_cs
2757       } } { \l_tmpa_tl } $
2758   }
2759 }
2760 }
2761 }

```

```

2762 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2763 }
2764 }
2765
2766 \cs_new:Nn \__stex_variables_reset:N {
2767 \tl_if_exist:NTF #1 {
2768 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2769 }{
2770 \let \exp_not:N #1 \exp_not:N \undefined
2771 }
2772 }
2773
2774 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2775 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2776 \exp_args:Nnx \use:nn {
2777 % TODO
2778 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2779 #2
2780 }
2781 }{
2782 \__stex_variables_reset:N \varnot
2783 \__stex_variables_reset:N \vartype
2784 \__stex_variables_reset:N \vardefi
2785 }
2786 }
2787
2788 \NewDocumentCommand \vardef { s } {
2789 \IfBooleanTF#1 {
2790 \__stex_variables_do_complex:nn
2791 }{
2792 \__stex_variables_do_simple:nnn
2793 }
2794 }
2795
2796 \NewDocumentCommand \svar { O{} m }{
2797 \tl_if_empty:nTF {#1}{
2798 \str_set:Nn \l_tmpa_str { #2 }
2799 }{
2800 \str_set:Nn \l_tmpa_str { #1 }
2801 }
2802 \__stex_term_omv:nn {
2803 var: // \l_tmpa_str
2804 }{ \comp{ #2 } }
2805 }
2806
2807 </package>

```

Chapter 31

STEX -Terms Implementation

```
2808 <*package>
2809
2810 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2811
2812 <@@=stex_terms>
2813
2814 Warnings and error messages
2815 \msg_new:nnn{stex}{error/nonotation}{
2816   Symbol~#1~invoked,~but~has~no~notation~#2!
2817 }
2818 \msg_new:nnn{stex}{error/notationarg}{
2819   Error~in~parsing~notation~#1
2820 }
2821 \msg_new:nnn{stex}{error/noop}{
2822   Symbol~#1~has~no~operator~notation~for~notation~#2
2823 }
2824 \msg_new:nnn{stex}{error/notallowed}{
2825   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2826 }
```

31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2826
2827 \cs_new:Nn \__stex_terms_reset:N {
2828   \tl_if_exist:NTF #1 {
2829     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2830   }{
2831     \let \exp_not:N #1 \exp_not:N \undefined
2832   }
2833 }
2834
2835 \bool_new:N \l_stex_allow_semantic_bool
2836 \bool_set_true:N \l_stex_allow_semantic_bool
```

```

2837
2838 \cs_new_protected:Nn \stex_invoke_symbol:n {
2839   \bool_if:NTF \l_stex_allow_semantic_bool {
2840     \str_if_eq:eeF {
2841       \prop_item:cn {
2842         l_stex_symdecl_#1_prop
2843       }{ deprecate }
2844     }{}{
2845       \msg_warning:nxxx{stex}{warning/deprecated}{
2846         Symbol~#1
2847       }{
2848         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2849       }
2850     }
2851     \if_mode_math:
2852       \exp_after:wN \__stex_terms_invoke_math:n
2853     \else:
2854       \exp_after:wN \__stex_terms_invoke_text:n
2855     \fi: { #1 }
2856   }{
2857     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2858   }
2859 }
2860
2861 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2862   \peek_charcode_remove:NTF ! {
2863     \__stex_terms_invoke_op_custom:nn {#1}
2864   }{
2865     \__stex_terms_invoke_custom:nn {#1}
2866   }
2867 }
2868
2869 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2870   \peek_charcode_remove:NTF ! {
2871     % operator
2872     \peek_charcode_remove:NTF * {
2873       % custom op
2874       \__stex_terms_invoke_op_custom:nn {#1}
2875     }{
2876       % op notation
2877       \peek_charcode:NTF [ {
2878         \__stex_terms_invoke_op_notation:nw {#1}
2879       }{
2880         \__stex_terms_invoke_op_notation:nw {#1}[]
2881       }
2882     }
2883   }{
2884     \peek_charcode_remove:NTF * {
2885       \__stex_terms_invoke_custom:nn {#1}
2886       % custom
2887     }{
2888       % normal
2889       \peek_charcode:NTF [ {
2890         \__stex_terms_invoke_notation:nw {#1}

```

```

2891     }{
2892       \__stex_terms_invoke_notation:nw {#1}[]
2893     }
2894   }
2895 }
2896 }
2897
2898
2899 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2900   \exp_args:Nnx \use:nn {
2901     \def\comp{\_comp}
2902     \str_set:Nn \l_stex_current_symbol_str { #1 }
2903     \bool_set_false:N \l_stex_allow_semantic_bool
2904     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2905       \comp{ #2 }
2906     }
2907   }{
2908     \__stex_terms_reset:N \comp
2909     \__stex_terms_reset:N \l_stex_current_symbol_str
2910     \bool_set_true:N \l_stex_allow_semantic_bool
2911   }
2912 }
2913
2914 \keys_define:nn { stex / terms } {
2915   lang .tl_set_x:N = \l_stex_notation_lang_str ,
2916   variant .tl_set_x:N = \l_stex_notation_variant_str ,
2917   unknown .code:n = \str_set:Nx
2918     \l_stex_notation_variant_str \l_keys_key_str
2919 }
2920
2921 \cs_new_protected:Nn \__stex_terms_args:n {
2922   \str_clear:N \l_stex_notation_lang_str
2923   \str_clear:N \l_stex_notation_variant_str
2924
2925   \keys_set:nn { stex / terms } { #1 }
2926 }
2927
2928 \cs_new_protected:Nn \stex_find_notation:nn {
2929   \__stex_terms_args:n { #2 }
2930   \seq_if_empty:cTF {
2931     l_stex_symdecl_ #1 _notations
2932   } {
2933     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2934   } {
2935     \bool_lazy_all:nTF {
2936       {\str_if_empty_p:N \l_stex_notation_variant_str}
2937       {\str_if_empty_p:N \l_stex_notation_lang_str}
2938     }{
2939       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
2940     }{
2941       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
2942         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
2943       }{
2944         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str

```

```

2945     }{
2946       \msg_error:nxxx{stex}{error/nonotation}{#1}{
2947         ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
2948       }
2949     }
2950   }
2951 }
2952 }
2953
2954 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
2955   \exp_args:Nnx \use:nn {
2956     \def\comp{\_comp}
2957     \str_set:Nn \l_stex_current_symbol_str { #1 }
2958     \stex_find_notation:nn { #1 }{ #2 }
2959     \bool_set_false:N \l_stex_allow_semantic_bool
2960     \cs_if_exist:cTF {
2961       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
2962     }{
2963       \_stex_term_oms:nnn {
2964         #1 \c_hash_str \l_stex_notation_variant_str
2965       }{ #1 }{
2966         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
2967       }
2968     }{
2969       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
2970         \cs_if_exist:cTF {
2971           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
2972         }{
2973           \tl_set:Nx \stex_symbol_after_invokation_tl {
2974             \__stex_terms_reset:N \comp
2975             \__stex_terms_reset:N \stex_symbol_after_invokation_tl
2976             \__stex_terms_reset:N \l_stex_current_symbol_str
2977             \bool_set_true:N \l_stex_allow_semantic_bool
2978           }
2979           \def\comp{\_comp}
2980           \str_set:Nn \l_stex_current_symbol_str { #1 }
2981           \bool_set_false:N \l_stex_allow_semantic_bool
2982           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
2983         }{
2984           \msg_error:nxxx{stex}{error/nonotation}{#1}{
2985             ~\l_stex_notation_variant_str
2986           }
2987         }
2988       }{
2989         \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
2990       }
2991     }
2992   }{
2993     \__stex_terms_reset:N \comp
2994     \__stex_terms_reset:N \l_stex_current_symbol_str
2995     \bool_set_true:N \l_stex_allow_semantic_bool
2996   }
2997 }
2998

```



```

2999 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3000   \stex_find_notation:nn { #1 }{ #2 }
3001   \cs_if_exist:cTF {
3002     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3003   }{
3004     \tl_set:Nx \stex_symbol_after_invokation_tl {
3005       \__stex_terms_reset:N \comp
3006       \__stex_terms_reset:N \stex_symbol_after_invokation_tl
3007       \__stex_terms_reset:N \l_stex_current_symbol_str
3008       \bool_set_true:N \l_stex_allow_semantic_bool
3009     }
3010     \def\comp{\_comp}
3011     \str_set:Nn \l_stex_current_symbol_str { #1 }
3012     \bool_set_false:N \l_stex_allow_semantic_bool
3013     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3014   }{
3015     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3016       ~\l_stex_notation_variant_str
3017     }
3018   }
3019 }
3020
3021 \prop_new:N \l__stex_terms_custom_args_prop
3022
3023 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3024   \exp_args:Nnx \use:nn {
3025     \bool_set_false:N \l_stex_allow_semantic_bool
3026     \def\comp{\_comp}
3027     \str_set:Nn \l_stex_current_symbol_str { #1 }
3028     \prop_clear:N \l__stex_terms_custom_args_prop
3029     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3030     \prop_get:cnN {
3031       l_stex_symdecl_#1 _prop
3032     }{ args } \l_tmpa_str
3033     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3034     \tl_set:Nn \arg { \__stex_terms_arg: }
3035     \str_if_empty:NTF \l_tmpa_str {
3036       \_stex_term_oms:nnn {#1}{#1}{#2}
3037     }{
3038       \str_if_in:NnTF \l_tmpa_str b {
3039         \_stex_term_ombind:nnn {#1}{#1}{#2}
3040       }{
3041         \str_if_in:NnTF \l_tmpa_str B {
3042           \_stex_term_ombind:nnn {#1}{#1}{#2}
3043         }{
3044           \_stex_term_oma:nnn {#1}{#1}{#2}
3045         }
3046       }
3047     }
3048     % TODO check that all arguments exist
3049   }{
3050     \__stex_terms_reset:N \l_stex_current_symbol_str
3051     \__stex_terms_reset:N \arg
3052     \__stex_terms_reset:N \comp

```

```

3053     \_stex_terms_reset:N \_stex_terms_custom_args_prop
3054     \bool_set_true:N \_stex_allow_semantic_bool
3055   }
3056 }
3057
3058 \NewDocumentCommand \_stex_terms_arg: { s O{} m}{
3059   \tl_if_empty:nTF {#2}{
3060     \int_set:Nn \l_tmpa_int {\prop_item:Nn \_stex_terms_custom_args_prop {currnum}}
3061     \bool_set_true:N \l_tmpa_bool
3062     \bool_do_while:Nn \l_tmpa_bool {
3063       \exp_args:NNx \prop_if_in:NnTF \_stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3064       \int_incr:N \l_tmpa_int
3065     }{
3066       \bool_set_false:N \l_tmpa_bool
3067     }
3068   }
3069   {
3070     \int_set:Nn \l_tmpa_int { #2 }
3071     \exp_args:NNx \prop_if_in:NnT \_stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3072       % TODO throw error
3073     }
3074   }
3075   \str_set:Nx \l_tmpa_str {\prop_item:Nn \_stex_terms_custom_args_prop {args} }
3076   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3077     % TODO throw error
3078   }
3079   \bool_set_true:N \_stex_allow_semantic_bool
3080   \IfBooleanTF#1{
3081     \stex_annotate_invisible:n {
3082       \exp_args:No \_stex_term_arg:nn {\_stex_current_symbol_str}{#3}
3083     }
3084   }{
3085     \exp_args:No \_stex_term_arg:nn {\_stex_current_symbol_str}{#3}
3086   }
3087   \bool_set_false:N \_stex_allow_semantic_bool
3088 }
3089
3090
3091 \cs_new_protected:Nn \_stex_term_arg:nn {
3092   \bool_set_true:N \_stex_allow_semantic_bool
3093   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3094   \bool_set_false:N \_stex_allow_semantic_bool
3095 }
3096
3097 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3098   \exp_args:Nnx \use:nn
3099   { \int_set:Nn \_stex_terms_downprec { #2 }
3100     \_stex_term_arg:nn { #1 }{ #3 }
3101   }
3102   { \int_set:Nn \exp_not:N \_stex_terms_downprec { \int_use:N \_stex_terms_downprec }
3103   }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3104 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3105   % TODO sequences
3106   \clist_set:Nn \l_tmpa_clist{ #3 }
3107   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3108     \tl_set:Nn \l_tmpa_tl { #3 }
3109   }{
3110     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3111     \clist_reverse:N \l_tmpa_clist
3112     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3113
3114     \clist_map_inline:Nn \l_tmpa_clist {
3115       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3116         \exp_args:Nno
3117         \l_tmpa_cs { ##1 } \l_tmpa_tl
3118       }
3119     }
3120   }
3121   \exp_args:Nnno
3122   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3123 }

```

(End definition for `_stex_term_math_assoc_arg:nnnn`. This function is documented on page 36.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3124 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3125 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3126 \int_new:N \l__stex_terms_downprec
3127 \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 37.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3128 \tl_set:Nn \l__stex_terms_left_bracket_str (
3129 \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

```

3130 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
3131   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3132     \bool_set_false:N \l__stex_terms_brackets_done_bool
3133     #2
3134   } {
3135     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3136       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3137         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3138         \dobrackets { #2 }

```

```

3139     }
3140     }{ #2 }
3141   }
3142 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

3143 \bool_new:N \l__stex_terms_brackets_done_bool
3144 %\RequirePackage{scalerel}
3145 \cs_new_protected:Npn \dobrackets #1 {
3146   %\ThisStyle{\if D@m@switch
3147   %   \exp_args:Nnx \use:nn
3148   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3149   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3150   % \else
3151   \exp_args:Nnx \use:nn
3152   {
3153     \bool_set_true:N \l__stex_terms_brackets_done_bool
3154     \int_set:Nn \l__stex_terms_downprec \infpref
3155     \l__stex_terms_left_bracket_str
3156     #1
3157   }
3158   {
3159     \bool_set_false:N \l__stex_terms_brackets_done_bool
3160     \l__stex_terms_right_bracket_str
3161     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3162   }
3163   %\fi}
3164 }

```

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

`\withbrackets`

```

3165 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3166   \exp_args:Nnx \use:nn
3167   {
3168     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3169     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3170     #3
3171   }
3172   {
3173     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3174     {\l__stex_terms_left_bracket_str}
3175     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3176     {\l__stex_terms_right_bracket_str}
3177   }
3178 }

```

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

`\STEXinvisible`

```

3179 \cs_new_protected:Npn \STEXinvisible #1 {
3180   \stex_annotate_invisible:n { #1 }
3181 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3182 \cs_new_protected:Nn \_stex_term_oms:nnn {
3183   \stex_annotate:nnn{ OMID }{ #2 }{
3184     \stex_highlight_term:nn { #1 } { #3 }
3185   }
3186 }
3187
3188 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3189   \__stex_terms_maybe_brackets:nn { #3 }{
3190     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3191   }
3192 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 36.)

`_stex_term_math_omv:nn`

```

3193 \cs_new_protected:Nn \_stex_term_omv:nn {
3194   \stex_annotate:nnn{ OMID }{ #1 }{
3195     \stex_highlight_term:nn { #1 } { #2 }
3196   }
3197 }

```

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

`_stex_term_math_oma:nnnn`

```

3198 \cs_new_protected:Nn \_stex_term_oma:nnn {
3199   \stex_annotate:nnn{ OMA }{ #2 }{
3200     \stex_highlight_term:nn { #1 } { #3 }
3201   }
3202 }
3203
3204 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3205   \__stex_terms_maybe_brackets:nn { #3 }{
3206     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3207   }
3208 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 36.)

`_stex_term_math_omb:nnnn`

```

3209 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3210   \stex_annotate:nnn{ OMBIND }{ #2 }{
3211     \stex_highlight_term:nn { #1 } { #3 }
3212   }
3213 }
3214
3215 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3216   \__stex_terms_maybe_brackets:nn { #3 }{
3217     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3218   }
3219 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 36.)

`\symref`
`\symname`

```

3220 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3221
3222 \keys_define:nn { stex / symname } {
3223   pre      .tl_set_x:N    = \l__stex_terms_pre_tl ,
3224   post     .tl_set_x:N    = \l__stex_terms_post_tl ,
3225   root     .tl_set_x:N    = \l__stex_terms_root_tl
3226 }
3227
3228 \cs_new_protected:Nn \stex_symname_args:n {
3229   \tl_clear:N \l__stex_terms_post_tl
3230   \tl_clear:N \l__stex_terms_pre_tl
3231   \tl_clear:N \l__stex_terms_root_str
3232   \keys_set:nn { stex / symname } { #1 }
3233 }
3234
3235 \NewDocumentCommand \symref { m m }{
3236   \let\compemph_uri_prev:\compemph@uri
3237   \let\compemph@uri\symrefemph@uri
3238   \STEXsymbol{#1}!\{ #2 }
3239   \let\compemph@uri\compemph_uri_prev:
3240 }
3241
3242 \NewDocumentCommand \synonym { 0{} m m }{
3243   \stex_symname_args:n { #1 }
3244   \let\compemph_uri_prev:\compemph@uri
3245   \let\compemph@uri\symrefemph@uri
3246   % TODO
3247   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3248   \let\compemph@uri\compemph_uri_prev:
3249 }
3250
3251 \NewDocumentCommand \symname { 0{} m }{
3252   \stex_symname_args:n { #1 }
3253   \stex_get_symbol:n { #2 }
3254   \str_set:Nx \l_tmpa_str {
3255     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3256   }
3257   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3258
3259   \let\compemph_uri_prev:\compemph@uri
3260   \let\compemph@uri\symrefemph@uri
3261   \exp_args:NNx \use:nn
3262   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3263     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3264   } }
3265   \let\compemph@uri\compemph_uri_prev:
3266 }
3267
3268 \NewDocumentCommand \Symname { 0{} m }{
3269   \stex_symname_args:n { #1 }
3270   \stex_get_symbol:n { #2 }

```

```

3271 \str_set:Nx \l_tmpa_str {
3272   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3273 }
3274 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3275 \let\compemph_uri_prev:\compemph@uri
3276 \let\compemph@uri\symrefemph@uri
3277 \exp_args:NNx \use:nn
3278 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3279   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3280   \l__stex_terms_post_tl
3281 } }
3282 \let\compemph@uri\compemph_uri_prev:
3283 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 36.)

31.3 Notation Components

```

3284 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3285 \cs_new_protected:Nn \stex_highlight_term:nn {
3286   #2
3287 }
3288
3289 \cs_new_protected:Nn \stex_unhighlight_term:n {
3290   % \latexml_if:TF {
3291   %   #1
3292   % } {
3293   %   \rustex_if:TF {
3294   %     #1
3295   %   } {
3296     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3297   % }
3298   % }
3299 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3300 \cs_new_protected:Npn \_comp #1 {
3301   \str_if_empty:NF \l_stex_current_symbol_str {
3302     \rustex_if:TF {
3303       \stex_annotate:nnn { comp } { \l_stex_current_symbol_str } { #1 }
3304     } {
3305       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3306     }
3307   }
3308 }
3309
3310 \cs_new_protected:Npn \_varcomp #1 {
3311   \str_if_empty:NF \l_stex_current_symbol_str {
3312     \rustex_if:TF {
3313       \stex_annotate:nnn { varcomp } { \l_stex_current_symbol_str } { #1 }

```

```

3314     }{
3315     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3316     }
3317   }
3318 }
3319
3320 \def\comp{\_comp}
3321
3322 \cs_new_protected:Npn \compemph@uri #1 #2 {
3323   \compemph{ #1 }
3324 }
3325
3326
3327 \cs_new_protected:Npn \compemph #1 {
3328   #1
3329 }
3330
3331 \cs_new_protected:Npn \defemph@uri #1 #2 {
3332   \defemph{#1}
3333 }
3334
3335 \cs_new_protected:Npn \defemph #1 {
3336   \textbf{#1}
3337 }
3338
3339 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3340   \symrefemph{#1}
3341 }
3342
3343 \cs_new_protected:Npn \symrefemph #1 {
3344   \textbf{#1}
3345 }
3346
3347 \cs_new_protected:Npn \varemp@uri #1 #2 {
3348   \varemp{#1}
3349 }
3350
3351 \cs_new_protected:Npn \varemp #1 {
3352   #1
3353 }

```

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

`\ellipses`

```

3354 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3355 \bool_new:N \l_stex_inarray_bool
\parrayline 3356 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3357 \NewDocumentCommand \parray { m m } {
\parraycell 3358   \begingroup
3359   \bool_set_true:N \l_stex_inarray_bool
3360   \begin{array}{#1}

```



```

3361     #2
3362   \end{array}
3363   \endgroup
3364 }
3365
3366 \NewDocumentCommand \prmatrix { m } {
3367   \begingroup
3368   \bool_set_true:N \l_stex_inarray_bool
3369   \begin{matrix}
3370     #1
3371   \end{matrix}
3372   \endgroup
3373 }
3374
3375 \def \maybepline {
3376   \bool_if:NT \l_stex_inarray_bool {\hline}
3377 }
3378
3379 \def \parrayline #1 #2 {
3380   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3381 }
3382
3383 \def \pmrow #1 { \parrayline{}{ #1 } }
3384
3385 \def \parraylineh #1 #2 {
3386   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3387 }
3388
3389 \def \parraycell #1 {
3390   #1 \bool_if:NT \l_stex_inarray_bool {&}
3391 }

```

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

31.4 Variables

```

3392 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3393 \cs_new_protected:Nn \stex_invoke_variable:n {
3394   \if_mode_math:
3395     \exp_after:wN \__stex_variables_invoke_math:n
3396   \else:
3397     \exp_after:wN \__stex_variables_invoke_text:n
3398   \fi: {#1}
3399 }
3400
3401 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3402   %TODO
3403 }
3404
3405
3406 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3407   \peek_charcode_remove:NTF ! {

```

```

3408 \peek_charcode_remove:NTF ! {
3409 \peek_charcode:NTF [ {
3410 \__stex_variables_invoke_op_custom:nw
3411 }{
3412 % TODO throw error
3413 }
3414 }{
3415 \__stex_variables_invoke_op:n { #1 }
3416 }
3417 }{
3418 \peek_charcode_remove:NTF * {
3419 \__stex_variables_invoke_text:n { #1 }
3420 }{
3421 \__stex_variables_invoke_math_ii:n { #1 }
3422 }
3423 }
3424 }
3425
3426 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3427 \cs_if_exist:cTF {
3428 stex_var_op_notation_ #1 _cs
3429 }{
3430 \exp_args:Nnx \use:nn {
3431 \def\comp{\_varcomp}
3432 \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3433 \_stex_term_omv:nn { var://#1 }{
3434 \use:c{stex_var_op_notation_ #1 _cs }
3435 }
3436 }{
3437 \__stex_variables_reset:N \comp
3438 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {\l_stex_current_symbol_str}
3439 }
3440 }{
3441 \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3442 \__stex_variables_invoke_math_ii:n {#1}
3443 }{
3444 \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
3445 }
3446 }
3447 }
3448
3449 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3450 \cs_if_exist:cTF {
3451 stex_var_notation_#1_cs
3452 }{
3453 \tl_set:Nx \stex_symbol_after_invocation_tl {
3454 \__stex_variables_reset:N \comp
3455 \__stex_variables_reset:N \stex_symbol_after_invocation_tl
3456 \__stex_variables_reset:N \l_stex_current_symbol_str
3457 \bool_set_true:N \l_stex_allow_semantic_bool
3458 }
3459 \def\comp{\_varcomp}
3460 \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3461 \bool_set_false:N \l_stex_allow_semantic_bool

```

```

3462     \use:c{stex_var_notation_#1_cs}
3463   }{
3464     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
3465   }
3466 }

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

3467 </package>

```

Chapter 32

STEX -Structural Features Implementation

```
3468 ⟨*package⟩
3469
3470 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3471
3472     Warnings and error messages
3473 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3474     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3475 }
3476 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3477     Symbol~#1~not~assigned~in~interpretmodule~#2
3478 }
3479 \msg_new:nnn{stex}{error/unknownstructure}{
3480     No~structure~#1~found!
3481 }
3482
3483 \msg_new:nnn{stex}{error/unknownfield}{
3484     No~field~#1~in~instance~#2~found!
3485 }
3486
3487 \msg_new:nnn{stex}{error/keyval}{
3488     Invalid~key=value~pair~#1
3489 }
3490 \msg_new:nnn{stex}{error/instantiate/missing}{
3491     Assignments~missing~in~instantiate:~#1
3492 }
3493 \msg_new:nnn{stex}{error/incompatible}{
3494     Incompatible~signature:~#1~(#2)~and~#3~(#4)
3495 }
3496
```

32.1 Imports with modification

```

3497 <@@=stex_copymodule>
3498 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3499   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3500     \tl_set:Nn \l_tmpa_tl { #1 }
3501     \__stex_copymodule_get_symbol_from_cs:
3502   }{
3503     % argument is a string
3504     % is it a command name?
3505     \cs_if_exist:cTF { #1 }{
3506       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3507       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3508       \str_if_empty:NTF \l_tmpa_str {
3509         \exp_args:Nx \cs_if_eq:NNTF {
3510           \tl_head:N \l_tmpa_tl
3511         } \stex_invoke_symbol:n {
3512           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3513         }{
3514           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3515         }
3516       } {
3517         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3518       }
3519     }{
3520       % argument is not a command name
3521       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3522       % \l_stex_all_symbols_seq
3523     }
3524   }
3525 }
3526
3527 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3528   \str_set:Nn \l_tmpa_str { #1 }
3529   \bool_set_false:N \l_tmpa_bool
3530   \bool_if:NF \l_tmpa_bool {
3531     \tl_set:Nn \l_tmpa_tl {
3532       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3533     }
3534     \str_set:Nn \l_tmpa_str { #1 }
3535     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3536     \seq_map_inline:Nn #2 {
3537       \str_set:Nn \l_tmpb_str { ##1 }
3538       \str_if_eq:eeT { \l_tmpa_str } {
3539         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3540       } {
3541         \seq_map_break:n {
3542           \tl_set:Nn \l_tmpa_tl {
3543             \str_set:Nn \l_stex_get_symbol_uri_str {
3544               ##1
3545             }
3546           }
3547         }
3548       }

```

```

3549     }
3550     \l_tmpa_tl
3551   }
3552 }
3553
3554 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3555   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3556     { \tl_tail:N \l_tmpa_tl }
3557   \tl_if_single:NTF \l_tmpa_tl {
3558     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3559       \exp_after:wN \str_set:Nn \exp_after:wN
3560         \l_stex_get_symbol_uri_str \l_tmpa_tl
3561       \__stex_copymodule_get_symbol_check:n { #1 }
3562     }{
3563       % TODO
3564       % tail is not a single group
3565     }
3566   }{
3567     % TODO
3568     % tail is not a single group
3569   }
3570 }
3571
3572 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3573   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3574     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3575       :~\seq_use:Nn #1 {,~}
3576     }
3577   }
3578 }
3579
3580 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3581   \stex_import_module_uri:nn { #1 } { #2 }
3582   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3583   \stex_import_require_module:nnnn
3584     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3585     { \l_stex_import_path_str } { \l_stex_import_name_str }
3586   \stex_collect_imports:n { \l_stex_import_ns_str ? \l_stex_import_name_str }
3587   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3588   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3589   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3590     \seq_map_inline:cn {c_stex_module_###1_constants}{
3591       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3592         ##1 ? ####1
3593       }
3594     }
3595   }
3596   \seq_clear:N \l_tmpa_seq
3597   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3598     name      = \l_stex_current_copymodule_name_str ,
3599     module    = \l_stex_current_module_str ,
3600     from      = \l_stex_import_ns_str ? \l_stex_import_name_str ,
3601     includes  = \l_tmpa_seq ,
3602     fields    = \l_tmpa_seq

```

```

3603 }
3604 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3605   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3606   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3607 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3608 \stex_if_smsmode:F {
3609   \begin{stex_annotate_env} {#4} {
3610     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3611   }
3612   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3613 }
3614 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3615 \bool_set_false:N \_stex_html_do_output_bool
3616 }
3617 \cs_new_protected:Nn \stex_copymodule_end:n {
3618   \def \l_tmpa_cs ##1 ##2 {#1}
3619   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3620   \tl_clear:N \l_tmpa_tl
3621   \tl_clear:N \l_tmpb_tl
3622   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3623   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3624     \seq_map_inline:cn {c_stex_module_##1_constants}{
3625       \tl_clear:N \l_tmpc_tl
3626       \l_tmpa_cs{##1}{####1}
3627       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3628         \tl_put_right:Nx \l_tmpa_tl {
3629           \prop_set_from_keyval:cn {
3630             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3631           }{
3632             \exp_after:wN \prop_to_keyval:N \csname
3633               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3634             \endcsname
3635           }
3636           \seq_clear:c {
3637             l_stex_symdecl_
3638             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3639             _notations
3640           }
3641         }
3642         \tl_put_right:Nx \l_tmpc_tl {
3643           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3644           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3645         }
3646         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3647       \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3648         \tl_put_right:Nx \l_tmpc_tl {
3649           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3650         }
3651         \tl_put_right:Nx \l_tmpa_tl {
3652           \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3653             \stex_invoke_symbol:n {
3654               \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3655             }
3656           }

```

```

3657     }
3658   }
3659 }{
3660   \tl_put_right:Nx \l_tmpc_tl {
3661     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3662   }
3663   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3664   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3665   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3666   \tl_put_right:Nx \l_tmpa_tl {
3667     \prop_set_from_keyval:cn {
3668       l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3669     }{
3670       \prop_to_keyval:N \l_tmpa_prop
3671     }
3672     \seq_clear:c {
3673       l_stex_symdecl_
3674       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3675       _notations
3676     }
3677   }
3678   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3679   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3680     \tl_put_right:Nx \l_tmpc_tl {
3681       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3682     }
3683     \tl_put_right:Nx \l_tmpa_tl {
3684       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3685         \stex_invoke_symbol:n {
3686           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3687         }
3688       }
3689     }
3690   }
3691 }
3692 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3693   \tl_put_right:Nx \l_tmpc_tl {
3694     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3695   }
3696 }
3697 \tl_put_right:Nx \l_tmpb_tl {
3698   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3699 }
3700 }
3701 }
3702 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3703 \tl_put_left:Nx \l_tmpa_tl {
3704   \prop_set_from_keyval:cn {
3705     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3706   }{
3707     \prop_to_keyval:N \l_stex_current_copymodule_prop
3708   }
3709 }
3710 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```



```

3711 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3712 \exp_args:Nx \stex_do_up_to_module:n {
3713   \exp_args:No \exp_not:n \l_tmpa_tl
3714 }
3715 \l_tmpb_tl
3716 \stex_if_smsmode:F {
3717   \end{stex_annotate_env}
3718 }
3719 }
3720
3721 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3722   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3723   \stex_deactivate_macro:Nn \symdecl {module~environments}
3724   \stex_deactivate_macro:Nn \symdef {module~environments}
3725   \stex_deactivate_macro:Nn \notation {module~environments}
3726   \stex_reactivate_macro:N \assign
3727   \stex_reactivate_macro:N \renamedec1
3728   \stex_reactivate_macro:N \donotcopy
3729   \stex_smsmode_do:
3730 }{
3731   \stex_copymodule_end:n {}
3732 }
3733
3734 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3735   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3736   \stex_deactivate_macro:Nn \symdecl {module~environments}
3737   \stex_deactivate_macro:Nn \symdef {module~environments}
3738   \stex_deactivate_macro:Nn \notation {module~environments}
3739   \stex_reactivate_macro:N \assign
3740   \stex_reactivate_macro:N \renamedec1
3741   \stex_reactivate_macro:N \donotcopy
3742   \stex_smsmode_do:
3743 }{
3744   \stex_copymodule_end:n {
3745     \tl_if_exist:cF {
3746       l__stex_copymodule_copymodule_##1?##2_def_tl
3747     }{
3748       \msg_error:nnxx{stex}{error/interpretmodule/nodefiniens}{
3749         ##1?##2
3750       }{\l_stex_current_copymodule_name_str}
3751     }
3752   }
3753 }
3754
3755 \NewDocumentCommand \donotcopy { 0{} m}{
3756   \stex_import_module_uri:nn { #1 } { #2 }
3757   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3758   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3759     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
3760     \seq_map_inline:cn {c_stex_module_##1_constants}{
3761       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3762       \bool_lazy_any_p:nT {
3763         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3764         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}

```

```

3765     { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3766   }{
3767     % TODO throw error
3768   }
3769 }
3770 }
3771
3772 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3773 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3774 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3775 }
3776
3777 \NewDocumentCommand \assign { m m }{
3778   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3779   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3780   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3781 }
3782
3783 \keys_define:nn { stex / renamedec1 } {
3784   name .str_set_x:N = \l_stex_renamedec1_name_str
3785 }
3786 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
3787   \str_clear:N \l_stex_renamedec1_name_str
3788   \keys_set:nn { stex / renamedec1 } { #1 }
3789 }
3790
3791 \NewDocumentCommand \renamedec1 { O{} m m }{
3792   \__stex_copymodule_renamedec1_args:n { #1 }
3793   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
3794   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3795   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3796   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3797     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3798       \l_stex_get_symbol_uri_str
3799     } }
3800   } {
3801     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
3802       \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3803       \prop_set_eq:cc {l_stex_symdecl_
3804         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3805         _prop
3806       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3807       \seq_set_eq:cc {l_stex_symdecl_
3808         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3809         _notations
3810       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3811       \prop_put:cnx {l_stex_symdecl_
3812         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3813         _prop
3814       }{ name }{ \l_stex_renamedec1_name_str }
3815       \prop_put:cnx {l_stex_symdecl_
3816         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3817         _prop
3818       }{ module }{ \l_stex_current_module_str }

```

```

3819 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
3820 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3821 }
3822 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3823 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
3824 } }
3825 }
3826 }
3827
3828 \stex_deactivate_macro:Nn \assign {copymodules}
3829 \stex_deactivate_macro:Nn \renameddecl {copymodules}
3830 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3831
3832
3833 \seq_new:N \l_stex_implicit_morphisms_seq
3834 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3835 \stex_import_module_uri:nn { #1 } { #2 }
3836 \stex_debug:nn{implicits}{
3837 Implicit~morphism:~
3838 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
3839 }
3840 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3841 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
3842 }{
3843 \msg_error:nnn{stex}{error/conflictingmodules}{
3844 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
3845 }
3846 }
3847
3848 % TODO
3849
3850
3851
3852 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3853 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
3854 }
3855 }
3856

```

32.2 The feature environment

structural@feature

```

3857 <@@=stex_features>
3858
3859 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
3860 \stex_if_in_module:F {
3861 \msg_set:nnn{stex}{error/nomodule}{
3862 Structural~Feature~has~to~occur~in~a~module:\\
3863 Feature~#2~of~type~#1\\
3864 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3865 }
3866 \msg_error:nn{stex}{error/nomodule}
3867 }

```

```

3868
3869 \stex_module_setup:nn{meta=NONE}{#2 - #1}
3870
3871 \stex_if_smsmode:F {
3872   \begin{stex_annotate_env}{ feature:#1 }{}
3873   \stex_annotate_invisible:nnn{header}{}{ #3 }
3874 }
3875 }{
3876   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
3877   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
3878   \stex_debug:nn{features}{
3879     Feature: \l_stex_last_feature_str
3880   }
3881   \stex_if_smsmode:F {
3882     \end{stex_annotate_env}
3883   }
3884 }

```

32.3 Structure

structure

```

3885 <@=stex_structures>
3886 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
3887   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
3888     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
3889   }
3890   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
3891     {#1}{#2}
3892 }
3893
3894 \keys_define:nn { stex / features / structure } {
3895   name .str_set_x:N = \l__stex_structures_name_str ,
3896 }
3897
3898 \cs_new_protected:Nn \__stex_structures_structure_args:n {
3899   \str_clear:N \l__stex_structures_name_str
3900   \keys_set:nn { stex / features / structure } { #1 }
3901 }
3902
3903 \NewDocumentEnvironment{mathstructure}{m O{}}{
3904   \__stex_structures_structure_args:n { #2 }
3905   \str_if_empty:NT \l__stex_structures_name_str {
3906     \str_set:Nx \l__stex_structures_name_str { #1 }
3907   }
3908   \exp_args:Nnnx
3909   \begin{structural_feature_module}{ structure }
3910     { \l__stex_structures_name_str }{}
3911   \stex_smsmode_do:
3912 }{
3913   \end{structural_feature_module}
3914   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
3915   \seq_clear:N \l_tmpa_seq
3916   \seq_map_inline:Nn \l_stex_collect_imports_seq {

```

```

3917     \seq_map_inline:cn{c_stex_module_##1_constants}{
3918       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
3919     }
3920   }
3921   \exp_args:Nnno
3922   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
3923   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
3924   \stex_add_structure_to_current_module:nn
3925     \l__stex_structures_name_str
3926     \l_stex_last_feature_str
3927   \exp_args:Nx \stex_symdecl_do:nn {
3928     name = \l__stex_structures_name_str ,
3929     type = \collection ,
3930     def = {\STEXsymbol{module-type}{
3931       \_stex_term_math_oms:nnnn { \l_stex_last_feature_str }{}{0}{}}
3932     }}
3933   }{ #1 }
3934   \exp_args:Nx
3935   \stex_add_to_current_module:n {
3936     \tl_set:cn { #1 }{
3937       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
3938     }
3939   }
3940   \exp_args:Nx
3941   \stex_do_up_to_module:n {
3942     \tl_set:cn { #1 }{
3943       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
3944     }
3945   }
3946 }
3947 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
3948
3949 \cs_new:Nn \stex_invoke_structure:nn {
3950   \stex_invoke_symbol:n { #1?#2 }
3951 }
3952
3953 \cs_new_protected:Nn \stex_get_structure:n {
3954   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3955     \tl_set:Nn \l_tmpa_tl { #1 }
3956     \__stex_structures_get_from_cs:
3957   }{
3958     \cs_if_exist:cTF { #1 }{
3959       \cs_set_eq:Nc \l_tmpa_cs { #1 }
3960       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
3961       \str_if_empty:NNTF \l_tmpa_str {
3962         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
3963           \__stex_structures_get_from_cs:
3964         }{
3965           \__stex_structures_get_from_string:n { #1 }
3966         }
3967       }{
3968         \__stex_structures_get_from_string:n { #1 }
3969       }
3970     }{

```

```

3971     \__stex_structures_get_from_string:n { #1 }
3972   }
3973 }
3974 }
3975
3976 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
3977   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3978     { \tl_tail:N \l_tmpa_tl }
3979   \str_set:Nx \l_tmpa_str {
3980     \exp_after:wN \use_i:nn \l_tmpa_tl
3981   }
3982   \str_set:Nx \l_tmpb_str {
3983     \exp_after:wN \use_ii:nn \l_tmpa_tl
3984   }
3985   \str_set:Nx \l_stex_get_structure_str {
3986     \l_tmpa_str ? \l_tmpb_str
3987   }
3988   \str_set:Nx \l_stex_get_structure_module_str {
3989     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
3990   }
3991 }
3992
3993 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
3994   \tl_set:Nn \l_tmpa_tl {
3995     \msg_error:nnn{stex}{error/unknownstructure}{#1}
3996   }
3997   \str_set:Nn \l_tmpa_str { #1 }
3998   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3999
4000   \seq_map_inline:Nn \l_stex_all_modules_seq {
4001     \prop_if_exist:cT {c_stex_module_##1_structures} {
4002       \prop_map_inline:cn {c_stex_module_##1_structures} {
4003         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4004           \prop_map_break:n{\seq_map_break:n{
4005             \tl_set:Nn \l_tmpa_tl {
4006               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4007               \str_set:Nn \l_stex_get_structure_module_str {####2}
4008             }
4009           }}
4010         }
4011       }
4012     }
4013   }
4014   \l_tmpa_tl
4015 }

```

\instantiate

```

4016
4017 \keys_define:nn { stex / instantiate } {
4018   name .str_set_x:N = \l__stex_structures_name_str
4019 }
4020 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4021   \str_clear:N \l__stex_structures_name_str
4022   \keys_set:nn { stex / instantiate } { #1 }

```

```

4023 }
4024
4025 \NewDocumentCommand \instantiate {m O{} m m}{
4026   \beginingroup
4027     \stex_get_structure:n {#4}
4028     \__stex_structures_instantiate_args:n { #2 }
4029     \str_if_empty:NT \l__stex_structures_name_str {
4030       \str_set:Nn \l__stex_structures_name_str { #1 }
4031     }
4032     \seq_clear:N \l__stex_structures_fields_seq
4033     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4034     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4035       \seq_map_inline:cn {c_stex_module_##1_constants}{
4036         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4037       }
4038     }
4039     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4040     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4041     \prop_clear:N \l_tmpa_prop
4042     \seq_map_inline:Nn \l_tmpa_seq {
4043       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4044       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4045         \msg_error:nnn{stex}{error/keyval}{##1}
4046       }
4047       \exp_args:Nx \stex_get_symbol_in_seq:nn { \seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4048       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4049       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4050       \exp_args:Nx \stex_get_symbol:n { \seq_item:Nn \l_tmpb_seq 2}
4051       \exp_args:Nxx \str_if_eq:nnF
4052         { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4053         { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4054         \msg_error:nnxxx{stex}{error/incompatible}
4055         { \l__stex_structures_dom_str
4056           { \prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4057           { \l_stex_get_symbol_uri_str
4058             { \prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4059           }
4060         }
4061       }
4062       \seq_if_empty:NF \l__stex_structures_fields_seq {
4063         \msg_error:nnx{stex}{error/instantiate/missing}{ \seq_use:Nn \l__stex_structures_fields_
4064       }
4065       \exp_args:Nx
4066       \stex_add_to_current_module:n {
4067         \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4068         domain = \l_stex_get_structure_module_str ,
4069         \prop_to_keyval:N \l_tmpa_prop
4070       }
4071       \tl_set:cn{ #1 }{ \stex_invoke_instance:nn{ \l_stex_current_module_str?\l__stex_structur
4072     }
4073     \exp_args:Nx
4074     \stex_do_up_to_module:n {
4075       \prop_set_from_keyval:cn {l_stex_instance\l_stex_current_module_str?\l__stex_structur
4076       domain = \l_stex_get_structure_module_str ,

```

```

4077     \prop_to_keyval:N \l_tmpa_prop
4078   }
4079   \tl_set:cn{ #1 }{\stex_invoke_instance:nn{\l_stex_current_module_str?\l_stex_structur
4080 }
4081 \exp_args:Nxx \stex_symdecl_do:nn {
4082   type={\STEXsymbol{module-type}}{
4083     \stex_term_math_oms:nnnn {
4084       \l_stex_get_structure_module_str
4085     }{}{0}{}
4086   }}
4087   }{\l_stex_structures_name_str}
4088 \endgroup
4089 \stex_smsmode_do:
4090 }
4091 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4092
4093 \cs_new_protected:Nn \stex_symbol_or_var:n {
4094   \cs_if_exist:cTF{#1}{
4095     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4096     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4097     \str_if_empty:NTF \l_tmpa_str {
4098       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4099       \stex_invoke_variable:n {
4100         \bool_set_true:N \l_stex_symbol_or_var_bool
4101         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4102         \str_set:Nx \l_stex_get_symbol_uri_str {
4103           \exp_after:wN \use:n \l_tmpa_tl
4104         }
4105       }{
4106         \bool_set_false:N \l_stex_symbol_or_var_bool
4107         \stex_get_symbol:n{#1}
4108       }
4109     }{
4110       \__stex_structures_symbolorvar_from_string:n{ #1 }
4111     }
4112   }{
4113     \__stex_structures_symbolorvar_from_string:n{ #1 }
4114   }
4115 }
4116
4117 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4118   \prop_if_exist:cTF {\l_stex_variable_#1 _prop}{
4119     \bool_set_true:N \l_stex_symbol_or_var_bool
4120     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4121   }{
4122     \bool_set_false:N \l_stex_symbol_or_var_bool
4123     \stex_get_symbol:n{#1}
4124   }
4125 }
4126
4127
4128 \NewDocumentCommand \varinstantiate {m O{} m m}{
4129   \begingroup
4130     \stex_get_structure:n {#4}

```



```

4131 \__stex_structures_instantiate_args:n { #2 }
4132 \str_if_empty:NT \l__stex_structures_name_str {
4133   \str_set:Nn \l__stex_structures_name_str { #1 }
4134 }
4135 \seq_clear:N \l__stex_structures_fields_seq
4136 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4137 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4138   \seq_map_inline:cn {c_stex_module_##1_constants}{
4139     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4140   }
4141 }
4142 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4143 \prop_clear:N \l_tmpa_prop
4144 \tl_if_empty:nF {#3} {
4145   \seq_set_split:Nnn \l_tmpa_seq , {#3}
4146   \seq_map_inline:Nn \l_tmpa_seq {
4147     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4148     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4149       \msg_error:nnn{stex}{error/keyval}{##1}
4150     }
4151     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4152     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4153     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4154     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4155     \bool_if:NTF \l_stex_symbol_or_var_bool {
4156       \exp_args:Nxx \str_if_eq:nnF
4157         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4158         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4159       \msg_error:nnxxx{stex}{error/incompatible}
4160         {\l__stex_structures_dom_str}
4161         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4162         {\l_stex_get_symbol_uri_str}
4163         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4164     }
4165     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4166   }}
4167   \exp_args:Nxx \str_if_eq:nnF
4168     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4169     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4170     \msg_error:nnxxx{stex}{error/incompatible}
4171     {\l__stex_structures_dom_str}
4172     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4173     {\l_stex_get_symbol_uri_str}
4174     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4175   }
4176   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4177 }
4178 }
4179 }
4180 \tl_gclear:N \g__stex_structures_aftergroup_tl
4181 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4182   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4183   \stex_find_notation:nn{##1}{}
4184   \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}

```

```

4185     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4186 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4187     \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}
4188     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4189 }
4190
4191 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4192     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4193         name = \l_tmpa_str ,
4194         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4195         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4196         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4197     }
4198     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4199     {g__stex_structures_tmpa_\l_tmpa_str _cs}
4200     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4201     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4202 }
4203 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invoke
4204 }
4205 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4206     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
4207         domain = \l_stex_get_structure_module_str ,
4208         \prop_to_keyval:N \l_tmpa_prop
4209     }
4210     \tl_set:cn { #1 }{\stex_invoke_varinstance:nn {\l__stex_structures_name_str}}
4211 }
4212 \aftergroup\g__stex_structures_aftergroup_tl
4213 \endgroup
4214 \stex_smsmode_do:
4215 }
4216
4217 \cs_new_protected:Nn \stex_invoke_instance:nn {
4218     \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4219         \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4220     }{
4221         \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4222     }
4223 }
4224
4225 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4226     \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4227         \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4228         \l_tmpa_tl
4229     }{
4230         \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}
4231     }
4232 }

```

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

\stex_invoke_structure:nnn

```

4233 % #1: URI of the instance
4234 % #2: URI of the instantiated module

```

```

4235 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4236   \tl_if_empty:nTF{ #3 }{
4237     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4238       c_stex_feature_ #2 _prop
4239     }
4240     \tl_clear:N \l_tmpa_tl
4241     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4242     \seq_map_inline:Nn \l_tmpa_seq {
4243       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4244       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4245       \cs_if_exist:cT {
4246         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4247       }{
4248         \tl_if_empty:NF \l_tmpa_tl {
4249           \tl_put_right:Nn \l_tmpa_tl {,}
4250         }
4251         \tl_put_right:Nx \l_tmpa_tl {
4252           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4253         }
4254       }
4255     }
4256     \exp_args:No \mathstruct \l_tmpa_tl
4257   }{
4258     \stex_invoke_symbol:n{#1/#3}
4259   }
4260 }

```

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

```

4261 </package>

```

Chapter 33

STEX -Statements Implementation

```
4262 <*package>
4263
4264 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4265
4266 <@@=stex_statements>
    Warnings and error messages
4267

\titleemph
4268 \def\titleemph#1{\textbf{#1}}

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

33.1 Definitions

definiendum

```
4269 \keys_define:nn {stex / definiendum }{
4270   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4271   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4272   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4273   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4274 }
4275 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4276   \str_clear:N \l__stex_statements_definiendum_root_str
4277   \tl_clear:N \l__stex_statements_definiendum_post_tl
4278   \str_clear:N \l__stex_statements_definiendum_gfa_str
4279   \keys_set:nn { stex / definiendum }{ #1 }
4280 }
4281 \NewDocumentCommand \definiendum { O{} m m } {
4282   \__stex_statements_definiendum_args:n { #1 }
4283   \stex_get_symbol:n { #2 }
4284   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4285   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4286     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4287     \tl_set:Nn \l_tmpa_tl { #3 }
4288   } {
4289     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4290     \tl_set:Nn \l_tmpa_tl {
4291       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4292     }
4293   }
4294 } {
4295   \tl_set:Nn \l_tmpa_tl { #3 }
4296 }
4297
4298 % TODO root
4299 \rustex_if:TF {
4300   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4301 } {
4302   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4303 }
4304 }
4305 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4306
4307 \NewDocumentCommand \definame { 0{ } m } {
4308   \__stex_statements_definiendum_args:n { #1 }
4309   % TODO: root
4310   \stex_get_symbol:n { #2 }
4311   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4312   \str_set:Nx \l_tmpa_str {
4313     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4314   }
4315   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4316   \rustex_if:TF {
4317     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4318       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4319     }
4320   } {
4321     \defemph@uri {
4322       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4323     } { \l_stex_get_symbol_uri_str }
4324   }
4325 }
4326 \stex_deactivate_macro:Nn \definame {definition~environments}
4327
4328 \NewDocumentCommand \Definame { 0{ } m } {
4329   \__stex_statements_definiendum_args:n { #1 }
4330   \stex_get_symbol:n { #2 }
4331   \str_set:Nx \l_tmpa_str {
4332     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4333   }
4334   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4335   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4336   \rustex_if:TF {

```

```

4337 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4338 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4339 }
4340 } {
4341 \defemph@uri {
4342 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4343 } { \l_stex_get_symbol_uri_str }
4344 }
4345 }
4346 \stex_deactivate_macro:Nn \Definame {definition~environments}
4347
4348 \NewDocumentCommand \premise { m }{
4349 \stex_annotate:nnn{ premise }{}{ #1 }
4350 }
4351 \NewDocumentCommand \conclusion { m }{
4352 \stex_annotate:nnn{ conclusion }{}{ #1 }
4353 }
4354 \NewDocumentCommand \definiens { m }{
4355 \stex_annotate:nnn{ definiens }{}{ #1 }
4356 }
4357
4358 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4359 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4360 \stex_deactivate_macro:Nn \definiens {definition~environments}
4361

```

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

sdefinition

```

4362
4363 \keys_define:nn {stex / sdefinition }{
4364 type .str_set_x:N = \sdefinitiontype,
4365 id .str_set_x:N = \sdefinitionid,
4366 name .str_set_x:N = \sdefinitionname,
4367 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4368 title .tl_set:N = \sdefinitiontitle
4369 }
4370 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4371 \str_clear:N \sdefinitiontype
4372 \str_clear:N \sdefinitionid
4373 \str_clear:N \sdefinitionname
4374 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4375 \tl_clear:N \sdefinitiontitle
4376 \keys_set:nn { stex / sdefinition }{ #1 }
4377 }
4378
4379 \NewDocumentEnvironment{sdefinition}{0{}}{
4380 \__stex_statements_sdefinition_args:n{ #1 }
4381 \stex_reactivate_macro:N \definiendum
4382 \stex_reactivate_macro:N \definame
4383 \stex_reactivate_macro:N \Definame
4384 \stex_reactivate_macro:N \premise
4385 \stex_reactivate_macro:N \definiens
4386 \stex_if_smsmode:F{

```

```

4387 \seq_clear:N \l_tmpa_seq
4388 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4389   \tl_if_empty:NF{ ##1 }{
4390     \stex_get_symbol:n { ##1 }
4391     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4392       \l_stex_get_symbol_uri_str
4393     }
4394   }
4395 }
4396 \exp_args:Nnnx
4397 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4398 \str_if_empty:NF \sdefinitiontype {
4399   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4400 }
4401 \clist_set:No \l_tmpa_clist \sdefinitiontype
4402 \tl_clear:N \l_tmpa_tl
4403 \clist_map_inline:Nn \l_tmpa_clist {
4404   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4405     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4406   }
4407 }
4408 \tl_if_empty:NTF \l_tmpa_tl {
4409   \__stex_statements_sdefinition_start:
4410 }{
4411   \l_tmpa_tl
4412 }
4413 }
4414 \stex_ref_new_doc_target:n \sdefinitionid
4415 \stex_smsmode_do:
4416 }{
4417   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4418   \stex_if_smsmode:F {
4419     \clist_set:No \l_tmpa_clist \sdefinitiontype
4420     \tl_clear:N \l_tmpa_tl
4421     \clist_map_inline:Nn \l_tmpa_clist {
4422       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4423         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4424       }
4425     }
4426     \tl_if_empty:NTF \l_tmpa_tl {
4427       \__stex_statements_sdefinition_end:
4428     }{
4429       \l_tmpa_tl
4430     }
4431     \end{stex_annotate_env}
4432   }
4433 }

```

\stexpatchdefinition

```

4434 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4435   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4436     ~(\sdefinitiontitle)
4437   }~}
4438 }

```

```

4439 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4440
4441 \newcommand\stexpatchdefinition[3] [] {
4442   \str_set:Nx \l_tmpa_str{ #1 }
4443   \str_if_empty:NTF \l_tmpa_str {
4444     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4445     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4446   }{
4447     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4448     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4449   }
4450 }

```

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

\inlinedef inline:

```

4451 \keys_define:nn {stex / inlinedef }{
4452   type      .str_set_x:N = \sdefinitiontype,
4453   id        .str_set_x:N = \sdefinitionid,
4454   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4455   name      .str_set_x:N = \sdefinitionname
4456 }
4457 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4458   \str_clear:N \sdefinitiontype
4459   \str_clear:N \sdefinitionid
4460   \str_clear:N \sdefinitionname
4461   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4462   \keys_set:nn { stex / inlinedef }{ #1 }
4463 }
4464 \NewDocumentCommand \inlinedef { 0{} m } {
4465   \beginngroup
4466   \__stex_statements_inlinedef_args:n{ #1 }
4467   \stex_reactivate_macro:N \definiendum
4468   \stex_reactivate_macro:N \definame
4469   \stex_reactivate_macro:N \Definame
4470   \stex_reactivate_macro:N \premise
4471   \stex_reactivate_macro:N \definiens
4472   \stex_ref_new_doc_target:n \sdefinitionid
4473   \stex_if_smsmode:TF{
4474     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4475   }{
4476     \seq_clear:N \l_tmpa_seq
4477     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4478       \tl_if_empty:nF{ ##1 }{
4479         \stex_get_symbol:n { ##1 }
4480         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4481           \l_stex_get_symbol_uri_str
4482         }
4483       }
4484     }
4485     \exp_args:Nnx
4486     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4487       \str_if_empty:NF \sdefinitiontype {
4488         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```



```

4489     }
4490     #2
4491     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4492   }
4493 }
4494 \endgroup
4495 \stex_smsmode_do:
4496 }

```

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

33.2 Assertions

`sassertion`

```

4497
4498 \keys_define:nn {stex / sassertion }{
4499   type      .str_set_x:N = \sassertiontype,
4500   id        .str_set_x:N = \sassertionid,
4501   title     .tl_set:N    = \sassertiontitle ,
4502   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4503   name      .str_set_x:N = \sassertionname
4504 }
4505 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4506   \str_clear:N \sassertiontype
4507   \str_clear:N \sassertionid
4508   \str_clear:N \sassertionname
4509   \clist_clear:N \l__stex_statements_sassertion_for_clist
4510   \tl_clear:N \sassertiontitle
4511   \keys_set:nn { stex / sassertion }{ #1 }
4512 }
4513
4514 %\tl_new:N \g__stex_statements_aftergroup_tl
4515
4516 \NewDocumentEnvironment{sassertion}{0{}}{
4517   \__stex_statements_sassertion_args:n{ #1 }
4518   \stex_reactivate_macro:N \premise
4519   \stex_reactivate_macro:N \conclusion
4520   \stex_if_smsmode:F {
4521     \seq_clear:N \l_tmpa_seq
4522     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4523       \tl_if_empty:nF{ ##1 }{
4524         \stex_get_symbol:n { ##1 }
4525         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4526           \l_stex_get_symbol_uri_str
4527         }
4528       }
4529     }
4530     \exp_args:Nnnx
4531     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4532     \str_if_empty:NF \sassertiontype {
4533       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4534     }
4535     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4536 \tl_clear:N \l_tmpa_tl
4537 \clist_map_inline:Nn \l_tmpa_clist {
4538   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4539     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4540   }
4541 }
4542 \tl_if_empty:NTF \l_tmpa_tl {
4543   \__stex_statements_sassertion_start:
4544 }{
4545   \l_tmpa_tl
4546 }
4547 }
4548 \str_if_empty:NTF \sassertionid {
4549   \str_if_empty:NF \sassertionname {
4550     \stex_ref_new_doc_target:n {}
4551   }
4552 } {
4553   \stex_ref_new_doc_target:n \sassertionid
4554 }
4555 \stex_smsmode_do:
4556 ){
4557   \str_if_empty:NF \sassertionname {
4558     \stex_symdecl_do:nn{ }\sassertionname}
4559   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4560 }
4561 \stex_if_smsmode:F {
4562   \clist_set:Nn \l_tmpa_clist \sassertiontype
4563   \tl_clear:N \l_tmpa_tl
4564   \clist_map_inline:Nn \l_tmpa_clist {
4565     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4566       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4567     }
4568   }
4569   \tl_if_empty:NTF \l_tmpa_tl {
4570     \__stex_statements_sassertion_end:
4571   }{
4572     \l_tmpa_tl
4573   }
4574   \end{stex_annotate_env}
4575 }
4576 }

```

\stexpatchassertion

```

4577
4578 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4579   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4580     (\sassertiontitle)
4581   }~}
4582 }
4583 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4584
4585 \newcommand\stexpatchassertion[3] [] {
4586   \str_set:Nx \l_tmpa_str{ #1 }
4587   \str_if_empty:NTF \l_tmpa_str {

```

```

4588     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4589     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4590   }{
4591     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4592     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4593   }
4594 }

```

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

\inlineass inline:

```

4595 \keys_define:nn {stex / inlineass }{
4596   type      .str_set_x:N = \sassertiontype,
4597   id        .str_set_x:N = \sassertionid,
4598   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4599   name      .str_set_x:N = \sassertionname
4600 }
4601 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4602   \str_clear:N \sassertiontype
4603   \str_clear:N \sassertionid
4604   \str_clear:N \sassertionname
4605   \clist_clear:N \l__stex_statements_sassertion_for_clist
4606   \keys_set:nn { stex / inlineass }{ #1 }
4607 }
4608 \NewDocumentCommand \inlineass { 0{} m } {
4609   \begin{group}
4610     \stex_reactivate_macro:N \premise
4611     \stex_reactivate_macro:N \conclusion
4612     \__stex_statements_inlineass_args:n{ #1 }
4613     \str_if_empty:NTF \sassertionid {
4614       \str_if_empty:NF \sassertionname {
4615         \stex_ref_new_doc_target:n { }
4616       }
4617     } {
4618       \stex_ref_new_doc_target:n \sassertionid
4619     }
4620
4621     \stex_if_smsmode:TF{
4622       \str_if_empty:NF \sassertionname {
4623         \stex_symdecl_do:nn{}{\sassertionname}
4624         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4625       }
4626     }{
4627       \seq_clear:N \l_tmpa_seq
4628       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4629         \tl_if_empty:nF{ ##1 }{
4630           \stex_get_symbol:n { ##1 }
4631           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4632             \l_stex_get_symbol_uri_str
4633           }
4634         }
4635       }
4636       \exp_args:Nnx
4637       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4638     \str_if_empty:NF \sassertiontype {
4639         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4640     }
4641     #2
4642     \str_if_empty:NF \sassertionname {
4643         \stex_symdecl_do:nn{}{\sassertionname}
4644         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4645     }
4646 }
4647 }
4648 \endgroup
4649 \stex_smsmode_do:
4650 }

```

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

33.3 Examples

`sexample`

```

4651
4652 \keys_define:nn {stex / sexample }{
4653     type      .str_set_x:N = \exampletype,
4654     id        .str_set_x:N = \sexampleid,
4655     title     .tl_set:N   = \sexampletitle,
4656     for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4657 }
4658 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4659     \str_clear:N \sexampletype
4660     \str_clear:N \sexampleid
4661     \tl_clear:N \sexampletitle
4662     \clist_clear:N \l__stex_statements_sexample_for_clist
4663     \keys_set:nn { stex / sexample }{ #1 }
4664 }
4665
4666 \NewDocumentEnvironment{sexample}{0{}}{
4667     \__stex_statements_sexample_args:n{ #1 }
4668     \stex_reactivate_macro:N \premise
4669     \stex_reactivate_macro:N \conclusion
4670     \stex_if_smsmode:F {
4671         \seq_clear:N \l_tmpa_seq
4672         \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4673             \tl_if_empty:nF{ ##1 }{
4674                 \stex_get_symbol:n { ##1 }
4675                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4676                     \l_stex_get_symbol_uri_str
4677                 }
4678             }
4679         }
4680         \exp_args:Nnnx
4681         \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4682         \str_if_empty:NF \sexampletype {
4683             \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
4684     }

```

```

4685 \clist_set:No \l_tmpa_clist \sexamplotype
4686 \tl_clear:N \l_tmpa_tl
4687 \clist_map_inline:Nn \l_tmpa_clist {
4688   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4689     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4690   }
4691 }
4692 \tl_if_empty:NTF \l_tmpa_tl {
4693   \__stex_statements_sexample_start:
4694 }{
4695   \l_tmpa_tl
4696 }
4697 }
4698 \str_if_empty:NF \sexampleid {
4699   \stex_ref_new_doc_target:n \sexampleid
4700 }
4701 \stex_smsmode_do:
4702 }{
4703   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4704   \stex_if_smsmode:F {
4705     \clist_set:No \l_tmpa_clist \sexamplotype
4706     \tl_clear:N \l_tmpa_tl
4707     \clist_map_inline:Nn \l_tmpa_clist {
4708       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4709         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4710       }
4711     }
4712     \tl_if_empty:NTF \l_tmpa_tl {
4713       \__stex_statements_sexample_end:
4714     }{
4715       \l_tmpa_tl
4716     }
4717     \end{stex_annotate_env}
4718   }
4719 }

```

\stexpatchexample

```

4720 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4721   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4722     (\sexamplename)
4723   }~}
4724 }
4725 }
4726 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4727
4728 \newcommand\stexpatchexample[3]{} {
4729   \str_set:Nx \l_tmpa_str{ #1 }
4730   \str_if_empty:NTF \l_tmpa_str {
4731     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4732     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4733   }{
4734     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4735     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4736   }

```

4737 }

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

`\inlineex` inline:

```

4738 \keys_define:nn {stex / inlineex }{
4739   type      .str_set_x:N = \sexamplotype,
4740   id        .str_set_x:N = \sexampleid,
4741   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4742   name      .str_set_x:N = \sexamplename
4743 }
4744 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4745   \str_clear:N \sexamplotype
4746   \str_clear:N \sexampleid
4747   \str_clear:N \sexamplename
4748   \clist_clear:N \l__stex_statements_sexample_for_clist
4749   \keys_set:nn { stex / inlineex }{ #1 }
4750 }
4751 \NewDocumentCommand \inlineex { 0{ } m } {
4752   \begingroup
4753   \stex_reactivate_macro:N \premise
4754   \stex_reactivate_macro:N \conclusion
4755   \__stex_statements_inlineex_args:n{ #1 }
4756   \str_if_empty:NF \sexampleid {
4757     \stex_ref_new_doc_target:n \sexampleid
4758   }
4759   \stex_if_smsmode:TF{
4760     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4761   }{
4762     \seq_clear:N \l_tmpa_seq
4763     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4764       \tl_if_empty:nF{ ##1 }{
4765         \stex_get_symbol:n { ##1 }
4766         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4767           \l_stex_get_symbol_uri_str
4768         }
4769       }
4770     }
4771     \exp_args:Nnx
4772     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4773       \str_if_empty:NF \sexamplotype {
4774         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4775       }
4776       #2
4777       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4778     }
4779   }
4780   \endgroup
4781   \stex_smsmode_do:
4782 }
```

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

33.4 Logical Paragraphs

sparagraph

```

4783 \keys_define:nn { stex / sparagraph } {
4784   id       .str_set_x:N = \sparagraphid ,
4785   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4786   type     .str_set_x:N = \sparagraphtype ,
4787   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4788   from     .tl_set:N    = \sparagraphfrom ,
4789   to       .tl_set:N    = \sparagraphto ,
4790   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4791   name     .str_set:N   = \sparagraphname
4792 }
4793
4794 \cs_new_protected:Nn \stex_sparagraph_args:n {
4795   \tl_clear:N \l_stex_sparagraph_title_tl
4796   \tl_clear:N \sparagraphfrom
4797   \tl_clear:N \sparagraphto
4798   \tl_clear:N \l_stex_sparagraph_start_tl
4799   \str_clear:N \sparagraphid
4800   \str_clear:N \sparagraphtype
4801   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4802   \str_clear:N \sparagraphname
4803   \keys_set:nn { stex / sparagraph } { #1 }
4804 }
4805 \newif\if@in@omtext\@in@omtextfalse
4806
4807 \NewDocumentEnvironment {sparagraph} { 0{} } {
4808   \stex_sparagraph_args:n { #1 }
4809   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4810     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4811   }{
4812     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4813   }
4814   \@in@omtexttrue
4815   \stex_if_smsmode:F {
4816     \seq_clear:N \l_tmpa_seq
4817     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4818       \tl_if_empty:NF{ ##1 }{
4819         \stex_get_symbol:n { ##1 }
4820         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4821           \l_stex_get_symbol_uri_str
4822         }
4823       }
4824     }
4825     \exp_args:Nnnx
4826     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4827     \str_if_empty:NF \sparagraphtype {
4828       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4829     }
4830     \str_if_empty:NF \sparagraphfrom {
4831       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4832     }
4833     \str_if_empty:NF \sparagraphto {

```

```

4834     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4835 }
4836 \clist_set:No \l_tmpa_clist \sparagraphtype
4837 \tl_clear:N \l_tmpa_tl
4838 \clist_map_inline:Nn \sparagraphtype {
4839     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4840         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4841     }
4842 }
4843 \tl_if_empty:NTF \l_tmpa_tl {
4844     \__stex_statements_sparagraph_start:
4845 }{
4846     \l_tmpa_tl
4847 }
4848 }
4849 \clist_set:No \l_tmpa_clist \sparagraphtype
4850 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4851     {
4852         \stex_reactivate_macro:N \definiendum
4853         \stex_reactivate_macro:N \definame
4854         \stex_reactivate_macro:N \Definame
4855         \stex_reactivate_macro:N \premise
4856         \stex_reactivate_macro:N \definiens
4857     }
4858     \str_if_empty:NTF \sparagraphid {
4859         \str_if_empty:NTF \sparagraphname {
4860             \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4861                 \stex_ref_new_doc_target:n {}
4862             }
4863         } {
4864             \stex_ref_new_doc_target:n {}
4865         }
4866     } {
4867         \stex_ref_new_doc_target:n \sparagraphid
4868     }
4869     \exp_args:NNx
4870     \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4871         \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4872             \tl_if_empty:nF{ ##1 }{
4873                 \stex_get_symbol:n { ##1 }
4874                 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4875             }
4876         }
4877     }
4878     \stex_smsmode_do:
4879     \ignorespacesandpars
4880 }{
4881     \str_if_empty:NF \sparagraphname {
4882         \stex_symdecl_do:nn{}{\sparagraphname}
4883         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4884     }
4885     \stex_if_smsmode:F {
4886         \clist_set:No \l_tmpa_clist \sparagraphtype
4887         \tl_clear:N \l_tmpa_tl

```



```

4888 \clist_map_inline:Nn \l_tmpa_clist {
4889   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4890     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4891   }
4892 }
4893 \tl_if_empty:NTF \l_tmpa_tl {
4894   \__stex_statements_sparagraph_end:
4895 }{
4896   \l_tmpa_tl
4897 }
4898 \end{stex_annotate_env}
4899 }
4900 }

```

\stexpatchparagraph

```

4901
4902 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4903   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4904     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4905       \titleemph{\l_stex_sparagraph_title_tl}:~
4906     }
4907   }{
4908     \titleemph{\l_stex_sparagraph_start_tl}~
4909   }
4910 }
4911 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4912
4913 \newcommand\stexpatchparagraph[3]{} {
4914   \str_set:Nx \l_tmpa_str{ #1 }
4915   \str_if_empty:NTF \l_tmpa_str {
4916     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4917     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4918   }{
4919     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4920     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4921   }
4922 }
4923
4924 \keys_define:nn { stex / inlinepara } {
4925   id      .str_set_x:N = \sparagraphid ,
4926   type    .str_set_x:N = \sparagraphtype ,
4927   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4928   from    .tl_set:N    = \sparagraphfrom ,
4929   to      .tl_set:N    = \sparagraphto ,
4930   name    .str_set:N    = \sparagraphname
4931 }
4932 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4933   \tl_clear:N \sparagraphfrom
4934   \tl_clear:N \sparagraphto
4935   \str_clear:N \sparagraphid
4936   \str_clear:N \sparagraphtype
4937   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4938   \str_clear:N \sparagraphname
4939   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

4940 }
4941 \NewDocumentCommand \inlinepara { 0{} m } {
4942   \begingroup
4943   \_stex_statements_inlinepara_args:n{ #1 }
4944   \clist_set:No \l_tmpa_clist \sparagraphtype
4945   \str_if_empty:NTF \sparaagraphid {
4946     \str_if_empty:NTF \sparaagraphname {
4947       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4948         \stex_ref_new_doc_target:n {}
4949       }
4950     } {
4951       \stex_ref_new_doc_target:n {}
4952     }
4953   } {
4954     \stex_ref_new_doc_target:n \sparaagraphid
4955   }
4956   \stex_if_smsmode:TF{
4957     \str_if_empty:NF \sparaagraphname {
4958       \stex_symdecl_do:nn{}{\sparaagraphname}
4959       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
4960     }
4961   }{
4962     \seq_clear:N \l_tmpa_seq
4963     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
4964       \tl_if_empty:nF{ ##1 }{
4965         \stex_get_symbol:n { ##1 }
4966         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4967           \l_stex_get_symbol_uri_str
4968         }
4969       }
4970     }
4971     \exp_args:Nnx
4972     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4973       \str_if_empty:NF \sparaagraphtype {
4974         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
4975       }
4976       \str_if_empty:NF \sparaagraphfrom {
4977         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
4978       }
4979       \str_if_empty:NF \sparaagraphto {
4980         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
4981       }
4982       \str_if_empty:NF \sparaagraphname {
4983         \stex_symdecl_do:nn{}{\sparaagraphname}
4984         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
4985       }
4986       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4987         \clist_map_inline:Nn \l_tmpa_seq {
4988           \stex_ref_new_sym_target:n {##1}
4989         }
4990       }
4991     } #2
4992   }
4993 }

```

```

4994 \endgroup
4995 \stex_smsmode_do:
4996 }
4997
(End definition for \stexpatchparagraph. This function is documented on page ??.)
4998 </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

```
4999 <*package>
5000 <@@=stex_sproof>
5001
5002 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5003
```

34.2 Proofs

We first define some keys for the proof environment.

```
5004 \keys_define:nn { stex / spf } {
5005   id          .str_set_x:N = \spfid,
5006   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5007   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5008   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5009   type        .str_set_x:N = \spftype,
5010   title       .tl_set:N    = \spftitle,
5011   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5012   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5013   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5014 }
5015 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5016   \str_clear:N \spfid
5017   \tl_clear:N \l__stex_sproof_spf_for_tl
5018   \tl_clear:N \l__stex_sproof_spf_from_tl
5019   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5020   \str_clear:N \spftype
5021   \tl_clear:N \spftitle
5022   \tl_clear:N \l__stex_sproof_spf_continues_tl
5023   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

5024 \tl_clear:N \l__stex_sproof_spf_method_tl
5025 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5026 \keys_set:nn { stex / spf }{ #1 }
5027 }

```

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

```

5028 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

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

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

```

5029 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5030 \cs_new_protected:Npn \sproofnumber {
5031   \int_set:Nn \l_tmpa_int {1}
5032   \bool_while_do:nn {
5033     \int_compare_p:nNn {
5034       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5035     } > 0
5036   }{
5037     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5038     \int_incr:N \l_tmpa_int
5039   }
5040 }
5041 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5042   \int_set:Nn \l_tmpa_int {1}
5043   \bool_while_do:nn {
5044     \int_compare_p:nNn {
5045       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5046     } > 0
5047   }{
5048     \int_incr:N \l_tmpa_int
5049   }
5050   \int_compare:nNnF \l_tmpa_int = 1 {
5051     \int_decr:N \l_tmpa_int
5052   }
5053   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5054     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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

```

5055 }
5056 }
5057
5058 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5059   \int_set:Nn \l_tmpa_int {1}
5060   \bool_while_do:nn {
5061     \int_compare_p:nNn {
5062       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5063     } > 0
5064   }{
5065     \int_incr:N \l_tmpa_int
5066   }
5067   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5068 }
5069
5070 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5071   \int_set:Nn \l_tmpa_int {1}
5072   \bool_while_do:nn {
5073     \int_compare_p:nNn {
5074       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5075     } > 0
5076   }{
5077     \int_incr:N \l_tmpa_int
5078   }
5079   \int_decr:N \l_tmpa_int
5080   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5081 }

```

\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

```

5082 \def\sproof@box{
5083   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5084 }
5085 \def\sproofend{
5086   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5087     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5088   }
5089 }

```

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

spf@*@kw

```

5090 \def\spf@proofsketch@kw{Proof-Sketch}
5091 \def\spf@proof@kw{Proof}
5092 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5093 \AddToHook{begindocument}{
5094   \ltx@ifpackageloaded{babel}{
5095     \makeatletter
5096     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5097     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5098       \input{sproof-ngerman.ldf}

```

```

5099     }
5100     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5101       \input{sproof-finnish.ldf}
5102     }
5103     \clist_if_in:NnT \l_tmpa_clist {french}{
5104       \input{sproof-french.ldf}
5105     }
5106     \clist_if_in:NnT \l_tmpa_clist {russian}{
5107       \input{sproof-russian.ldf}
5108     }
5109     \makeatother
5110   }{}
5111 }

```

spfsketch

```

5112 \newcommand\spfsketch[2] [] {
5113   \beginingroup
5114   \let \premise \stex_proof_premise:
5115   \__stex_sproof_spf_args:n{#1}
5116   \stex_if_smsmode:TF {
5117     \str_if_empty:NF \spfid {
5118       \stex_ref_new_doc_target:n \spfid
5119     }
5120   }{
5121     \seq_clear:N \l_tmpa_seq
5122     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5123       \tl_if_empty:NF{ ##1 }{
5124         \stex_get_symbol:n { ##1 }
5125         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5126           \l_stex_get_symbol_uri_str
5127         }
5128       }
5129     }
5130     \exp_args:Nnx
5131     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5132       \str_if_empty:NF \spftype {
5133         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5134       }
5135       \clist_set:No \l_tmpa_clist \spftype
5136       \tl_set:Nn \l_tmpa_tl {
5137         \titleemph{
5138           \tl_if_empty:NTF \spftitle {
5139             \spf@proofsketch@kw
5140           }{
5141             \spftitle
5142           }
5143         }::~
5144       }
5145       \clist_map_inline:Nn \l_tmpa_clist {
5146         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5147           \tl_clear:N \l_tmpa_tl
5148         }
5149       }
5150       \str_if_empty:NF \spfid {

```

```

5151         \stex_ref_new_doc_target:n \spfid
5152     }
5153     \l_tmpa_tl #2 \sproofend
5154 }
5155 }
5156 \endgroup
5157 \stex_smsmode_do:
5158 }
5159

```

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

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

```

5160 \newenvironment{spfeq}[2][]{
5161   \__stex_sproof_spf_args:n{#1}
5162   \let \premise \stex_proof_premise:
5163   \stex_if_smsmode:TF {
5164     \str_if_empty:NF \spfid {
5165       \stex_ref_new_doc_target:n \spfid
5166     }
5167   }{
5168     \seq_clear:N \l_tmpa_seq
5169     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5170       \tl_if_empty:NF{ ##1 }{
5171         \stex_get_symbol:n { ##1 }
5172         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5173           \l_stex_get_symbol_uri_str
5174         }
5175       }
5176     }
5177     \exp_args:Nnnx
5178     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5179     \str_if_empty:NF \spftype {
5180       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5181     }
5182
5183     \clist_set:No \l_tmpa_clist \spftype
5184     \tl_clear:N \l_tmpa_tl
5185     \clist_map_inline:Nn \l_tmpa_clist {
5186       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5187         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5188       }
5189       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5190         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5191       }
5192     }
5193     \tl_if_empty:NTF \l_tmpa_tl {
5194       \__stex_sproof_spfeq_start:
5195     }{
5196       \l_tmpa_tl
5197     }{-#2}

```

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

¹⁵EDNOTE: document above


```

5198 \str_if_empty:NF \spfid {
5199 \stex_ref_new_doc_target:n \spfid
5200 }
5201 \begin{displaymath}\begin{array}{rc1l}
5202 }
5203 \stex_smsmode_do:
5204 }{
5205 \stex_if_smsmode:F {
5206 \end{array}\end{displaymath}
5207 \clist_set:No \l_tmpa_clist \spftype
5208 \tl_clear:N \l_tmpa_tl
5209 \clist_map_inline:Nn \l_tmpa_clist {
5210 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5211 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5212 }
5213 }
5214 \tl_if_empty:NTF \l_tmpa_tl {
5215 \__stex_sproof_spfeq_end:
5216 }{
5217 \l_tmpa_tl
5218 }
5219 \end{stex_annotate_env}
5220 }
5221 }
5222
5223 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5224 \titleemph{
5225 \tl_if_empty:NTF \spftitle {
5226 \spf@proof@kw
5227 }{
5228 \spftitle
5229 }
5230 }:
5231 }
5232 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5233
5234 \newcommand\stexpatchspfeq[3] [] {
5235 \str_set:Nx \l_tmpa_str{ #1 }
5236 \str_if_empty:NTF \l_tmpa_str {
5237 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5238 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5239 }{
5240 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5241 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5242 }
5243 }
5244

```

(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.

```

5245 \newenvironment{sproof}[2] []{

```

```

5246 \let \premise \stex_proof_premise:
5247 \intarray_gzero:N \l__stex_sproof_counter_intarray
5248 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5249 \__stex_sproof_spf_args:n{#1}
5250 \stex_if_smsmode:TF {
5251   \str_if_empty:NF \spfid {
5252     \stex_ref_new_doc_target:n \spfid
5253   }
5254 }{
5255   \seq_clear:N \l_tmpa_seq
5256   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5257     \tl_if_empty:NF{ ##1 }{
5258       \stex_get_symbol:n { ##1 }
5259       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5260         \l_stex_get_symbol_uri_str
5261       }
5262     }
5263   }
5264   \exp_args:Nnnx
5265   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5266   \str_if_empty:NF \spftype {
5267     \stex_annotate_invisible:nnn{type}{\spftype}{}}
5268   }
5269
5270   \clist_set:No \l_tmpa_clist \spftype
5271   \tl_clear:N \l_tmpa_tl
5272   \clist_map_inline:Nn \l_tmpa_clist {
5273     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5274       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5275     }
5276     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5277       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5278     }
5279   }
5280   \tl_if_empty:NTF \l_tmpa_tl {
5281     \__stex_sproof_sproof_start:
5282   }{
5283     \l_tmpa_tl
5284   }{~#2}
5285   \str_if_empty:NF \spfid {
5286     \stex_ref_new_doc_target:n \spfid
5287   }
5288   \begin{description}
5289   }
5290   \stex_smsmode_do:
5291 }{
5292   \stex_if_smsmode:F{
5293     \end{description}
5294     \clist_set:No \l_tmpa_clist \spftype
5295     \tl_clear:N \l_tmpa_tl
5296     \clist_map_inline:Nn \l_tmpa_clist {
5297       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5298         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5299       }

```

```

5300     }
5301     \tl_if_empty:NTF \l_tmpa_tl {
5302         \__stex_sproof_sproof_end:
5303     }{
5304         \l_tmpa_tl
5305     }
5306     \end{stex_annotate_env}
5307 }
5308 }
5309
5310 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5311     \par\noindent\titleemph{
5312         \tl_if_empty:NTF \spftype {
5313             \spf@proof@kw
5314         }{
5315             \spftype
5316         }
5317     }:
5318 }
5319 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5320
5321 \newcommand\stexpatchsproof[3] [] {
5322     \str_set:Nx \l_tmpa_str{ #1 }
5323     \str_if_empty:NTF \l_tmpa_str {
5324         \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5325         \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5326     }{
5327         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5328         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5329     }
5330 }

```

\spfidea

```

5331 \newcommand\spfidea[2] []{
5332     \__stex_sproof_spf_args:n{#1}
5333     \titleemph{
5334         \tl_if_empty:NTF \spftype {Proof~Idea}{
5335             \spftype
5336         }:
5337     }~#2
5338     \sproofend
5339 }

```

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

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

spfstep

```

5340 \newenvironment{spfstep}[1] []{
5341     \__stex_sproof_spf_args:n{#1}
5342     \stex_if_smsmode:TF {

```

```

5343 \str_if_empty:NF \spfid {
5344 \stex_ref_new_doc_target:n \spfid
5345 }
5346 }{
5347 \in@contexttrue
5348 \seq_clear:N \l_tmpa_seq
5349 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5350 \tl_if_empty:NF{ ##1 }{
5351 \stex_get_symbol:n { ##1 }
5352 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5353 \l_stex_get_symbol_uri_str
5354 }
5355 }
5356 }
5357 \exp_args:Nnnx
5358 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5359 \str_if_empty:NF \spftype {
5360 \stex_annotate_invisible:nnn{type}{\spftype}{}
5361 }
5362 \clist_set:No \l_tmpa_clist \spftype
5363 \tl_set:Nn \l_tmpa_tl {
5364 \item[\sproofnumber]
5365 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5366 }
5367 \clist_map_inline:Nn \l_tmpa_clist {
5368 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5369 \tl_clear:N \l_tmpa_tl
5370 }
5371 }
5372 \l_tmpa_tl
5373 \tl_if_empty:NF \spftitle {
5374 {(\titleemph{\spftitle})\enspace}
5375 }
5376 \str_if_empty:NF \spfid {
5377 \stex_ref_new_doc_target:n \spfid
5378 }
5379 }
5380 \stex_smsmode_do:
5381 \ignorespacesandpars
5382 }{
5383 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5384 \__stex_sproof_inc_counter:
5385 }
5386 \stex_if_smsmode:F {
5387 \end{stex_annotate_env}
5388 }
5389 }

```

sproofcomment

```

5390 \newenvironment{sproofcomment}[1][]{
5391 \__stex_sproof_spf_args:n{#1}
5392 \clist_set:No \l_tmpa_clist \spftype
5393 \tl_set:Nn \l_tmpa_tl {
5394 \item[\sproofnumber]

```

```

5395 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5396 }
5397 \clist_map_inline:Nn \l_tmpa_clist {
5398   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5399     \tl_clear:N \l_tmpa_tl
5400   }
5401 }
5402 \l_tmpa_tl
5403 }{
5404   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5405     \__stex_sproof_inc_counter:
5406   }
5407 }

```

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

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

```

5408 \newenvironment{subproof}[2][]{
5409   \__stex_sproof_spf_args:n{#1}
5410   \stex_if_smsmode:TF{
5411     \str_if_empty:NF \spfid {
5412       \stex_ref_new_doc_target:n \spfid
5413     }
5414   }{
5415     \seq_clear:N \l_tmpa_seq
5416     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5417       \tl_if_empty:NF{ ##1 }{
5418         \stex_get_symbol:n { ##1 }
5419         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5420           \l_stex_get_symbol_uri_str
5421         }
5422       }
5423     }
5424     \exp_args:Nnnx
5425     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5426     \str_if_empty:NF \spftype {
5427       \stex_annotate_invisible:nnn{type}{\spftype}{}
5428     }
5429
5430     \clist_set:No \l_tmpa_clist \spftype
5431     \tl_set:Nn \l_tmpa_tl {
5432       \item[\sproofnumber]
5433       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5434     }
5435     \clist_map_inline:Nn \l_tmpa_clist {
5436       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5437         \tl_clear:N \l_tmpa_tl
5438       }
5439     }
5440     \l_tmpa_tl
5441     \tl_if_empty:NF \spftitle {
5442       {(\titleemph{\spftitle})\enspace}
5443     }

```

```

5444     {~#2}
5445     \str_if_empty:NF \spfid {
5446       \stex_ref_new_doc_target:n \spfid
5447     }
5448   }
5449   \__stex_sproof_add_counter:
5450   \stex_smsmode_do:
5451 }{
5452   \__stex_sproof_remove_counter:
5453   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5454     \__stex_sproof_inc_counter:
5455   }
5456   \stex_if_smsmode:F{
5457     \end{stex_annotate_env}
5458   }
5459 }

```

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

```

5460 \newenvironment{spfcases}[2][]{
5461   \tl_if_empty:nTF{#1}{
5462     \begin{subproof}[method=by-cases]{#2}
5463   }{
5464     \begin{subproof}[#1,method=by-cases]{#2}
5465   }
5466 }{
5467   \end{subproof}
5468 }

```

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

```

5469 \newenvironment{spfcase}[2][]{
5470   \__stex_sproof_spf_args:n{#1}
5471   \stex_if_smsmode:TF {
5472     \str_if_empty:NF \spfid {
5473       \stex_ref_new_doc_target:n \spfid
5474     }
5475   }{
5476     \seq_clear:N \l_tmpa_seq
5477     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5478       \tl_if_empty:nF{ ##1 }{
5479         \stex_get_symbol:n { ##1 }
5480         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5481           \l_stex_get_symbol_uri_str
5482         }
5483       }
5484     }
5485     \exp_args:Nnnx
5486     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5487     \str_if_empty:NF \spftype {
5488       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5489   }
5490   \clist_set:Nn \l_tmpa_clist \spftype
5491   \tl_set:Nn \l_tmpa_tl {
5492     \item[\sproofnumber]

```

```

5493     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5494   }
5495   \clist_map_inline:Nn \l_tmpa_clist {
5496     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5497       \tl_clear:N \l_tmpa_tl
5498     }
5499   }
5500   \l_tmpa_tl
5501   \tl_if_empty:nF{#2}{
5502     \titleemph{#2}:~
5503   }
5504 }
5505 \__stex_sproof_add_counter:
5506 \stex_smsmode_do:
5507 ){
5508   \__stex_sproof_remove_counter:
5509   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5510     \__stex_sproof_inc_counter:
5511   }
5512   \stex_if_smsmode:F{
5513     \clist_set:No \l_tmpa_clist \spftype
5514     \tl_set:Nn \l_tmpa_tl{\sproofend}
5515     \clist_map_inline:Nn \l_tmpa_clist {
5516       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5517         \tl_clear:N \l_tmpa_tl
5518       }
5519     }
5520     \l_tmpa_tl
5521     \end{stex_annotate_env}
5522   }
5523 }

```

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

```

5524 \newcommand\spfcasesketch[3][]{
5525   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5526 }

```

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.

```

5527 \keys_define:nn { stex / just }{
5528   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5529   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5530   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5531   args     .tl_set:N    = \l__stex_sproof_just_args_tl
5532 }

```

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

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

justification

```
5533 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5534 \newcommand\stex_proof_promise:[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.

```
5535 \newcommand\justarg[2] [] {#2}
```

```
5536 \end{package}
```

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

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

Chapter 35

STEX -Others Implementation

```
5537 <*package>
5538
5539 %%%%%%%%%% others.dtx %%%%%%%%%%
5540
5541 <@@=stex_others>
    Warnings and error messages
5542 % None

\MSC Math subject classifier

5543 \NewDocumentCommand \MSC {m} {
5544 % TODO
5545 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5546 \@ifpackageloaded{tikzinput}{
5547 \RequirePackage{stex-tikzinput}
5548 }{}
5549 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
5550 <*package>
5551 <@@=stex_modules>
5552
5553 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5554
5555 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5556 \begingroup
5557 \stex_module_setup:nn{
5558   ns=\c_stex_metatheory_ns_str,
5559   meta=NONE
5560 }{Metatheory}
5561 \stex_reactivate_macro:N \symdecl
5562 \stex_reactivate_macro:N \notation
5563 \stex_reactivate_macro:N \symdef
5564 \ExplSyntaxOff
5565 \csname stex_suppress_html:n\endcsname{
5566   % is-a (a:A, a \in A, a is an A, etc.)
5567   \symdecl{isa}[args=ai]
5568   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5569   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5570   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5571
5572   % bind (\forall, \Pi, \lambda etc.)
5573   \symdecl{bind}[args=Bi]
5574   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5575   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5576   \notation{bind}[depfun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5577
5578   % implicit bind
5579   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5580
5581   % dummy variable
5582   \symdecl{dummyvar}
5583   \notation{dummyvar}[underscore]{\comp\_}
5584   \notation{dummyvar}[dot]{\comp\cdot}
```

```

5585 \notation{dummyvar}[dash]{\comp{\rm --}}
5586
5587 %fromto (function space, Hom-set, implication etc.)
5588 \symdecl{fromto}[args=ai]
5589 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5590 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5591
5592 % mapto (lambda etc.)
5593 \symdecl{mapto}[args=Bi]
5594 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5595 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5596 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5597
5598 % function/operator application
5599 \symdecl{apply}[args=ia]
5600 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5601 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5602
5603 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5604 \symdecl{metacollection}
5605 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5606 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5607
5608 % collection of propositions/booleans/truth values
5609 \symdecl{prop}[name=proposition]
5610 \notation{prop}[prop]{\comp{\rm prop}}
5611 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5612
5613 % sequences
5614 \symdecl{seqtype}[args=1]
5615 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5616
5617 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5618 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{\comp\ast}}
5619
5620 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5621 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5622 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5623
5624 % letin (‘‘let’’, local definitions, variable substitution)
5625 \symdecl{letin}[args=bii]
5626 \notation{letin}[let]{\comp{\rm let}};#1\comp{=}#2;\comp{\rm in}};#3}
5627 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5628 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5629
5630 % structures
5631 \symdecl*{module-type}[args=1]
5632 \notation{module-type}{\mathtt{MOD} #1}
5633 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5634 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5635
5636 }
5637 \ExplSyntaxOn
5638 \stex_add_to_current_module:n{

```

```

5639 \let\nappa\apply
5640 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5641 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5642 \def\livar{\csname sequence-index\endcsname[li]}
5643 \def\uivar{\csname sequence-index\endcsname[ui]}
5644 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5645 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5646 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5647 }
5648 \__stex_modules_end_module:
5649 \endgroup
5650 </package>

```

Chapter 37

Tikzinput Implementation

```
5651 <*package>
5652
5653 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5654
5655 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5656 \RequirePackage{l3keys2e}
5657
5658 \keys_define:nn { tikzinput } {
5659   image .bool_set:N = \c_tikzinput_image_bool,
5660   image .default:n = false ,
5661   unknown .code:n = {}
5662 }
5663
5664 \ProcessKeysOptions { tikzinput }
5665
5666 \bool_if:NTF \c_tikzinput_image_bool {
5667   \RequirePackage{graphicx}
5668
5669   \providecommand\usetikzlibrary[]{}
5670   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
5671 }{
5672   \RequirePackage{tikz}
5673   \RequirePackage{standalone}
5674
5675   \newcommand \tikzinput [2] [] {
5676     \setkeys{Gin}{#1}
5677     \ifx \Gin@ewidth \Gin@exclamation
5678       \ifx \Gin@eheight \Gin@exclamation
5679         \input { #2 }
5680       \else
5681         \resizebox{!}{ \Gin@eheight }{
5682           \input { #2 }
5683         }
5684       \fi
5685     \else
5686       \ifx \Gin@eheight \Gin@exclamation
5687         \resizebox{ \Gin@ewidth }{!}{
5688           \input { #2 }
```

```

5689     }
5690     \else
5691         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5692             \input { #2 }
5693         }
5694     \fi
5695 \fi
5696 }
5697 }
5698
5699 \newcommand \ctikzinput [2] [] {
5700     \begin{center}
5701         \tikzinput [1] {#2}
5702     \end{center}
5703 }
5704
5705 \@ifpackageloaded{stex}{
5706     \RequirePackage{stex-tikzinput}
5707 }{}
5708
5709 </package>
5710 <*stex>
5711 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5712 \RequirePackage{stex}
5713 \RequirePackage{tikzinput}
5714
5715 \newcommand\mhtikzinput [2] [] {%
5716     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5717     \stex_in_repository:nn\Gin@mhrepos{
5718         \tikzinput [1]{\mhpath{##1}{#2}}
5719     }
5720 }
5721 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5722 </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.

```
5723 \*cls)
5724 \@@=document_structure)
5725 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5726 \RequirePackage{13keys2e}
```

38.2 Class Options

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

`\omdoc@cls@class`

```
5727 \keys_define:nn{ document-structure / pkg }{
5728   class      .str_set_x:N = \c_document_structure_class_str,
5729   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5730   report     .code:n      = {
5731     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5732     \str_set:Nn \c_document_structure_class_str {report}
5733   },
5734   book       .code:n      = {
5735     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5736     \str_set:Nn \c_document_structure_class_str {book}
5737   },
5738   bookpart   .code:n      = {
5739     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5740     \str_set:Nn \c_document_structure_class_str {book}
5741     \str_set:Nn \c_document_structure_topsect_str {chapter}
5742   },
```

```

5743 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5744 unknown     .code:n      = {
5745   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5746 }
5747 }
5748 \ProcessKeysOptions{ document-structure / pkg }
5749 \str_if_empty:NT \c_document_structure_class_str {
5750   \str_set:Nn \c_document_structure_class_str {article}
5751 }
5752 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5753   {\c_document_structure_class_str}
5754

```

38.3 Beefing up the document environment

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

```

5755 \RequirePackage{document-structure}
5756 \bool_if:NF \c_document_structure_minimal_bool {

```

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

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

```

5757 \keys_define:nn { document-structure / document }{
5758   id .str_set_x:N = \c_document_structure_document_id_str
5759 }
5760 \let\__document_structure_orig_document=\document
5761 \renewcommand{\document}[1][]{
5762   \keys_set:nn{ document-structure / document }{ #1 }
5763   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5764   \__document_structure_orig_document
5765 }

```

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

```

5766 }
5767 \</cls>

```

38.4 Implementation: document-structure Package

```

5768 <*package>
5769 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
5770 \RequirePackage{l3keys2e}

```

38.5 Package Options

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

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


```

5771
5772 \keys_define:nn{ document-structure / pkg }{
5773   class      .str_set_x:N = \c_document_structure_class_str,
5774   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5775   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5776 }
5777 \ProcessKeysOptions{ document-structure / pkg }
5778 \str_if_empty:NT \c_document_structure_class_str {
5779   \str_set:Nn \c_document_structure_class_str {article}
5780 }
5781 \str_if_empty:NT \c_document_structure_topsect_str {
5782   \str_set:Nn \c_document_structure_topsect_str {section}
5783 }

```

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

```

5784 \RequirePackage{xspace}
5785 \RequirePackage{comment}
5786 \AddToHook{begindocument}{
5787   \ltx@ifpackageloaded{babel}{
5788     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5789     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5790       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
5791     }
5792   }{}
5793 }

```

`\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.

```

5794 \int_new:N \l_document_structure_section_level_int
5795 \str_case:VnF \c_document_structure_topsect_str {
5796   {part}}{
5797     \int_set:Nn \l_document_structure_section_level_int {0}
5798   }
5799   {chapter}}{
5800     \int_set:Nn \l_document_structure_section_level_int {1}
5801   }
5802 }{
5803   \str_case:VnF \c_document_structure_class_str {
5804     {book}}{
5805       \int_set:Nn \l_document_structure_section_level_int {0}
5806     }
5807     {report}}{
5808       \int_set:Nn \l_document_structure_section_level_int {0}
5809     }
5810   }{
5811     \int_set:Nn \l_document_structure_section_level_int {2}
5812   }
5813 }

```

38.6 Document Structure

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

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁸

EdN:18

```
5814 \def\current@section@level{document}%
5815 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5816 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5817 \cs_new_protected:Npn \skipomgroup {
5818   \ifcase\l_document_structure_section_level_int
5819   \or\stepcounter{part}
5820   \or\stepcounter{chapter}
5821   \or\stepcounter{section}
5822   \or\stepcounter{subsection}
5823   \or\stepcounter{subsubsection}
5824   \or\stepcounter{paragraph}
5825   \or\stepcounter{subparagraph}
5826   \fi
5827 }
```

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

`blindfragment`

```
5828 \newcommand\at@begin@blindomgroup[1]{%
5829 \newenvironment{blindfragment}
5830 {
5831   \int_incr:N\l_document_structure_section_level_int
5832   \at@begin@blindomgroup\l_document_structure_section_level_int
5833 }{}}
```

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

```
5834 \newcommand\omgroup@nonum[2]{%
5835   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5836   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5837 }
```

(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.

```
5838 \newcommand\omgroup@num[2]{%
```

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

```

5839 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5840   \@nameuse{#1}{#2}
5841 }{
5842   \cs_if_exist:NTF\rdfmata@sectioning{
5843     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5844   }{
5845     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5846   }
5847 }
5848 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5849 }

```

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

sfragment

```

5850 \keys_define:nn { document-structure / omgroup }{
5851   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5852   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5853   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5854   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5855   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5856   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5857   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5858   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5859   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5860   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5861 }
5862 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5863   \str_clear:N \l__document_structure_omgroup_id_str
5864   \str_clear:N \l__document_structure_omgroup_date_str
5865   \clist_clear:N \l__document_structure_omgroup_creators_clist
5866   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5867   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5868   \tl_clear:N \l__document_structure_omgroup_type_tl
5869   \tl_clear:N \l__document_structure_omgroup_short_tl
5870   \tl_clear:N \l__document_structure_omgroup_display_tl
5871   \tl_clear:N \l__document_structure_omgroup_intro_tl
5872   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5873   \keys_set:nn { document-structure / omgroup } { #1 }
5874 }

```

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.

```

5875 \newif\if@mainmatter\@mainmattertrue
5876 \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.

```

5877 \keys_define:nn { document-structure / sectioning }{
5878   name .str_set_x:N = \l__document_structure_sect_name_str ,
5879   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5880   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5881   clear .default:n = {true} ,
5882   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5883   num      .default:n      = {true}
5884 }
5885 \cs_new_protected:Nn \__document_structure_sect_args:n {
5886   \str_clear:N \l__document_structure_sect_name_str
5887   \str_clear:N \l__document_structure_sect_ref_str
5888   \bool_set_false:N \l__document_structure_sect_clear_bool
5889   \bool_set_false:N \l__document_structure_sect_num_bool
5890   \keys_set:nn { document-structure / sectioning } { #1 }
5891 }
5892 \newcommand\omdoc@sectioning[3][]{
5893   \__document_structure_sect_args:n {#1}
5894   \let\omdoc@sect@name\l__document_structure_sect_name_str
5895   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5896   \if@mainmatter% numbering not overridden by frontmatter, etc.
5897     \bool_if:NTF \l__document_structure_sect_num_bool {
5898       \omgroup@num{#2}{#3}
5899     }{
5900       \omgroup@nonum{#2}{#3}
5901     }
5902     \def\current@section@level{\omdoc@sect@name}
5903   \else
5904     \omgroup@nonum{#2}{#3}
5905   \fi
5906 }% 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.

```

5907 \newcommand\omgroup@redefine@addtocontents[1]{%
5908 %\edef\__document_structureimport{#1}%
5909 %\@for\@I:=\__document_structureimport\do{%
5910 %\edef\@path{\csname module@\@I @path\endcsname}%
5911 %\@ifundefined{tf@toc}\relax%
5912 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5913 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5914 %\def\addcontentsline##1##2##3{%
5915 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5916 %\else% hyperref.sty not loaded
5917 %\def\addcontentsline##1##2##3{%
5918 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5919 %\fi
5920 }% 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.

```

5921 \newenvironment{sfragment}[2][]{% keys, title
5922 {
5923   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

5924   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5925     \omgroup@redefine@addtocontents{
5926       \@ifundefined{module@id}\used@modules%

```

```

5927     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5928   }
5929 }

```

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

```

5930 \int_incr:N\l_document_structure_section_level_int
5931 \ifcase\l_document_structure_section_level_int
5932   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5933   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5934   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5935   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5936   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5937   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5938   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5939 \fi
5940 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5941 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5942   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5943 }
5944 }% for customization
5945 {}

```

and finally, we localize the sections

```

5946 \newcommand\omdoc@part@kw{Part}
5947 \newcommand\omdoc@chapter@kw{Chapter}
5948 \newcommand\omdoc@section@kw{Section}
5949 \newcommand\omdoc@subsection@kw{Subsection}
5950 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5951 \newcommand\omdoc@paragraph@kw{paragraph}
5952 \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`

```

5953 \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).

```

5954 \cs_if_exist:NTF\frontmatter{
5955   \let\__document_structure_orig_frontmatter\frontmatter
5956   \let\frontmatter\relax
5957 }{
5958   \tl_set:Nn\__document_structure_orig_frontmatter{
5959     \clearpage
5960     \@mainmatterfalse
5961     \pagenumbering{roman}

```

```

5962 }
5963 }
5964 \cs_if_exist:NTF\backmatter{
5965   \let\__document_structure_orig_backmatter\backmatter
5966   \let\backmatter\relax
5967 }{
5968   \tl_set:Nn\__document_structure_orig_backmatter{
5969     \clearpage
5970     \@mainmatterfalse
5971     \pagenumbering{roman}
5972   }
5973 }

```

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.

```

5974 \newenvironment{frontmatter}{
5975   \__document_structure_orig_frontmatter
5976 }{
5977   \cs_if_exist:NTF\mainmatter{
5978     \mainmatter
5979   }{
5980     \clearpage
5981     \@mainmattertrue
5982     \pagenumbering{arabic}
5983   }
5984 }

```

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

```

5985 \newenvironment{backmatter}{
5986   \__document_structure_orig_backmatter
5987 }{
5988   \cs_if_exist:NTF\mainmatter{
5989     \mainmatter
5990   }{
5991     \clearpage
5992     \@mainmattertrue
5993     \pagenumbering{arabic}
5994   }
5995 }

```

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

```

5996 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5997 \def \c__document_structure_document_str{document}
5998 \newcommand\afterprematurestop{}
5999 \def\prematurestop@endomgroup{
6000   \unless\ifx\@currenvir\c__document_structure_document_str
6001     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6002       \expandafter\prematurestop@endomgroup

```

```

6003 \fi
6004 }
6005 \providecommand\prematurestop{
6006   \message{Stopping~sTeX~processing~prematurely}
6007   \prematurestop@endumgroup
6008   \afterprematurestop
6009   \end{document}}
6010 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

6011 \RequirePackage{etoolbox}
6012 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6013 \newrobustcmd\useSGvar[1]{%
6014   \@ifundefined{sTeX@Gvar@#1}
6015   {\PackageError{document-structure}
6016     {The sTeX Global variable #1 is undefined}
6017     {set it with \protect\setSGvar}}
6018   \@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.

```

6019 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6020   \@ifundefined{sTeX@Gvar@#1}
6021   {\PackageError{document-structure}
6022     {The sTeX Global variable #1 is undefined}
6023     {set it with \protect\setSGvar}}
6024   {\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.

```
6025 \*cls)
6026 \@@=notesslides)
6027 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6028 \RequirePackage{13keys2e}
6029
6030 \keys_define:nn{notesslides / cls}{
6031   class .code:n = {
6032     \PassOptionsToClass{\CurrentOption}{document-structure}
6033     \str_if_eq:nnT{#1}{book}{
6034       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6035     }
6036     \str_if_eq:nnT{#1}{report}{
6037       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6038     }
6039   },
6040   notes .bool_set:N = \c__notesslides_notes_bool ,
6041   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6042   unknown .code:n = {
6043     \PassOptionsToClass{\CurrentOption}{document-structure}
6044     \PassOptionsToClass{\CurrentOption}{beamer}
6045     \PassOptionsToPackage{\CurrentOption}{notesslides}
6046   }
6047 }
6048 \ProcessKeysOptions{ notesslides / cls }
6049 \bool_if:NTF \c__notesslides_notes_bool {
6050   \PassOptionsToPackage{notes=true}{notesslides}
6051 }{
6052   \PassOptionsToPackage{notes=false}{notesslides}
6053 }
6054 \</cls)
```


now we do the same for the notesslides package.

```

6055 <*package>
6056 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6057 \RequirePackage{13keys2e}
6058
6059 \keys_define:nn{notesslides / pkg}{
6060   topsect      .str_set_x:N = \c__notesslides_topsect_str,
6061   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
6062   notes        .bool_set:N = \c__notesslides_notes_bool ,
6063   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6064   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6065   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
6066   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6067   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
6068   unknown      .code:n      = {
6069     \PassOptionsToClass{\CurrentOption}{stex}
6070     \PassOptionsToClass{\CurrentOption}{tikzinput}
6071   }
6072 }
6073 \ProcessKeysOptions{ notesslides / pkg }
6074 \newif\ifnotes
6075 \bool_if:NTF \c__notesslides_notes_bool {
6076   \notesttrue
6077 }{
6078   \notesfalse
6079 }
6080

```

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

```

6081 \str_if_empty:NTF \c__notesslides_topsect_str {
6082   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6083 }{
6084   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6085 }
6086 </package>

```

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

```

6087 <*cls>
6088 \bool_if:NTF \c__notesslides_notes_bool {
6089   \LoadClass{document-structure}
6090 }{
6091   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6092   \newcounter{Item}
6093   \newcounter{paragraph}
6094   \newcounter{subparagraph}
6095   \newcounter{Hfootnote}
6096   \RequirePackage{document-structure}
6097 }

```

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

```

6098 \RequirePackage{notesslides}
6099 </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.

```

6100 <*package>
6101 \bool_if:NT \c__notesslides_notes_bool {
6102   \RequirePackage{a4wide}
6103   \RequirePackage{marginnote}
6104   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6105   \RequirePackage{mdframed}
6106   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6107   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6108 }
6109 \RequirePackage{stex-tikzinput}
6110 \RequirePackage{etoolbox}
6111 \RequirePackage{amssymb}
6112 \RequirePackage{amsmath}
6113 \RequirePackage{comment}
6114 \RequirePackage{textcomp}
6115 \RequirePackage{url}
6116 \RequirePackage{graphicx}
6117 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.¹⁹

```

6118 \bool_if:NT \c__notesslides_notes_bool {
6119   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6120 }
6121
6122
6123 \NewDocumentCommand \libusetheme {0{} m} {
6124   \bool_if:NTF \c__notesslides_notes_bool {
6125     \libusepackage[#1]{beamernotestheme#2}
6126   }{
6127     \libusepackage[#1]{beamertheme#2}
6128   }
6129 }

```

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

```

6130 \newcounter{slide}
6131 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6132 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

¹⁹EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

6133 \bool_if:NTF \c__notesslides_notes_bool {
6134   \renewenvironment{note}{\ignorespaces}{}
6135 }{
6136   \excludecomment{note}
6137 }

```

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.

```

6138 \bool_if:NT \c__notesslides_notes_bool {
6139   \newlength{\slideframewidth}
6140   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6141 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6142   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6143     \bool_set_true:N #1
6144   }{
6145     \bool_set_false:N #1
6146   }
6147 }
6148 \keys_define:nn{notesslides / frame}{
6149   label .str_set_x:N = \l__notesslides_frame_label_str,
6150   allowframebreaks .code:n = {
6151     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6152   },
6153   allowdisplaybreaks .code:n = {
6154     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6155   },
6156   fragile .code:n = {
6157     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6158   },
6159   shrink .code:n = {
6160     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6161   },
6162   squeeze .code:n = {
6163     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6164   },
6165   t .code:n = {
6166     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6167   },
6168 }
6169 \cs_new_protected:Nn \__notesslides_frame_args:n {
6170   \str_clear:N \l__notesslides_frame_label_str
6171   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6172   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6173   \bool_set_true:N \l__notesslides_frame_fragile_bool
6174   \bool_set_true:N \l__notesslides_frame_shrink_bool
6175   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6176   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6177 \keys_set:nn { notesslides / frame }{ #1 }
6178 }

```

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

```

6179 \renewenvironment{frame}[1][]{
6180   \__notesslides_frame_args:n{#1}
6181   \sffamily
6182   \stepcounter{slide}
6183   \def\@currentlabel{\theslide}
6184   \str_if_empty:NF \l__notesslides_frame_label_str {
6185     \label{\l__notesslides_frame_label_str}
6186   }

```

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

```

6187 \def\itemize@level{outer}
6188 \def\itemize@outer{outer}
6189 \def\itemize@inner{inner}
6190 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6191 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6192 \renewenvironment{itemize}{
6193   \ifx\itemize@level\itemize@outer
6194     \def\itemize@label{$\rhd$}
6195   \fi
6196   \ifx\itemize@level\itemize@inner
6197     \def\itemize@label{$\scriptstyle\rhd$}
6198   \fi
6199   \begin{list}
6200     {\itemize@label}
6201     {\setlength{\labelsep}{.3em}
6202      \setlength{\labelwidth}{.5em}
6203      \setlength{\leftmargin}{1.5em}
6204     }
6205   \edef\itemize@level{\itemize@inner}
6206 }{
6207   \end{list}
6208 }

```

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

```

6209 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6210 }{
6211   \medskip\miko@slidelabel\end{mdframed}
6212 }

```

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

`\frametitle`

```

6213 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6214 }

```

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

EdN:20

`\pause`

```

20
6215 \bool_if:NT \c__notesslides_notes_bool {
6216   \newcommand\pause{}
6217 }

```

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

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

nparagraph

```
6218 \bool_if:NTF \c__notesslides_notes_bool {
6219   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6220 }{
6221   \excludecomment{nparagraph}
6222 }
```

nfragment

```
6223 \bool_if:NTF \c__notesslides_notes_bool {
6224   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6225 }{
6226   \excludecomment{nfragment}
6227 }
```

ndefinition

```
6228 \bool_if:NTF \c__notesslides_notes_bool {
6229   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6230 }{
6231   \excludecomment{ndefinition}
6232 }
```

nassertion

```
6233 \bool_if:NTF \c__notesslides_notes_bool {
6234   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6235 }{
6236   \excludecomment{nassertion}
6237 }
```

nsproof

```
6238 \bool_if:NTF \c__notesslides_notes_bool {
6239   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6240 }{
6241   \excludecomment{nsproof}
6242 }
```

nexample

```
6243 \bool_if:NTF \c__notesslides_notes_bool {
6244   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6245 }{
6246   \excludecomment{nexample}
6247 }
```

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

```
6248 \def\inputref@preskip{\smallskip}
6249 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6250 \let\orig@inputref\inputref
6251 \def\inputref{\@ifstar\ninputref\orig@inputref}
6252 \newcommand\ninputref[2][]{
6253   \bool_if:NT \c__notesslides_notes_bool {
6254     \orig@inputref[#1]{#2}
6255   }
6256 }
```

(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 \LaTeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
6257 \newlength{\slidelogoheight}
6258
6259 \bool_if:NTF \c__notesslides_notes_bool {
6260   \setlength{\slidelogoheight}{.4cm}
6261 }{
6262   \setlength{\slidelogoheight}{1cm}
6263 }
6264 \newsavebox{\slidelogo}
6265 \sbox{\slidelogo}{\text{\LaTeX}}
6266 \newrobustcmd{\setslidelogo}[1]{\def\slidelogo{#1}}
6267 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6268 }
```

(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.

```
6269 \def\source{Michael Kohlhase}% customize locally
6270 \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.

```
6271 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6272 \newsavebox{\cclogo}
6273 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6274 \newif\ifcchref\cchreffalse
6275 \AtBeginDocument{
6276   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6277 }
6278 \def\licensing{
6279   \ifcchref
```

```

6280     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6281   \else
6282     {\usebox{\cclogo}}
6283   \fi
6284 }
6285 \newrobustcmd{\setlicensing}[2][]{
6286   \def\@url{\#1}
6287   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
6288   \ifx\@url\@empty
6289     \def\licensing{\usebox{\cclogo}}
6290   \else
6291     \def\licensing{
6292       \ifcchref
6293         \href{\#1}{\usebox{\cclogo}}
6294       \else
6295         {\usebox{\cclogo}}
6296       \fi
6297     }
6298   \fi
6299 }

```

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

EdN:21

\slidelabel Now, we set up the slide label for the article mode.²¹

```

6300 \newrobustcmd\miko@slidelabel{
6301   \vbox to \slidelogoheight{
6302     \vss\hbox to \slidewidth
6303     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6304   }
6305 }

```

(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.

```

6306 \def\Gin@mhrepos{}
6307 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{\#1}}
6308 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
6309 \newrobustcmd\frameimage[2][]{
6310   \stepcounter{slide}
6311   \bool_if:NT \c__notesslides_frameimages_bool {
6312     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
6313     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6314     \begin{center}
6315       \bool_if:NTF \c__notesslides_fboxed_bool {
6316         \fbox{
6317           \ifx\Gin@ewidth\@empty
6318             \ifx\Gin@mhrepos\@empty
6319               \mhgraphics[width=\slidewidth,\#1]{\#2}
6320             \else

```

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

```

6321         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6322     \fi
6323 \else% Gin@ewidth empty
6324     \ifx\Gin@mhrepos\@empty
6325         \mhgraphics[#1]{#2}
6326     \else
6327         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6328     \fi
6329 \fi% Gin@ewidth empty
6330 }
6331 }{
6332     \ifx\Gin@ewidth\@empty
6333         \ifx\Gin@mhrepos\@empty
6334             \mhgraphics[width=\slidewidth,#1]{#2}
6335         \else
6336             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6337         \fi
6338         \ifx\Gin@mhrepos\@empty
6339             \mhgraphics[#1]{#2}
6340         \else
6341             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6342         \fi
6343     \fi% Gin@ewidth empty
6344 }
6345 \end{center}
6346 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6347 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6348 }
6349 } % 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.

```

6350 \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

```

6351 \AddToHook{begindocument}{
6352     \definecolor{green}{rgb}{0,.5,0}
6353     \definecolor{purple}{cmyk}{.3,1,0,.17}
6354 }

```

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`.

```

6355 % \def\STpresent#1{\textcolor{blue}{#1}}
6356 \def\defemph#1{\textcolor{magenta}{#1}}
6357 \def\symrefemph#1{\textcolor{cyan}{#1}}
6358 \def\compemph#1{\textcolor{blue}{#1}}
6359 \def\titleemph#1{\textcolor{blue}{#1}}
6360 \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.

```

6361 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6362 \def\smalltextwarning{
6363   \pgfuseimage{miko@small@dbend}
6364   \xspace
6365 }
6366 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6367 \newrobustcmd\textwarning{
6368   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6369   \xspace
6370 }
6371 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6372 \newrobustcmd\bigtextwarning{
6373   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6374   \xspace
6375 }

```

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

```

6376 \newrobustcmd\putgraphicsat[3]{
6377   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6378 }
6379 \newrobustcmd\putat[2]{
6380   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6381 }

```

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`.

```

6382 \bool_if:NT \c__notesslides_sectocframes_bool {
6383   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6384     \newcounter{chapter}\counterwithin*{section}{chapter}
6385   }{
6386     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6387       \newcounter{chapter}\counterwithin*{section}{chapter}
6388     }
6389   }
6390 }

```

`\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
6391 \def\part@prefix{}
6392 \@ifpackageloaded{document-structure}{}{
6393   \str_case:VnF \__notesslidesstopsect {
6394     {part}{
6395       \int_set:Nn \l_document_structure_section_level_int {0}
6396       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6397     \def\part@prefix{\arabic{chapter}.}
6398   }
6399   {chapter}{
6400     \int_set:Nn \l_document_structure_section_level_int {1}
6401     \def\thesection{\arabic{chapter}.\arabic{section}}
6402     \def\part@prefix{\arabic{chapter}.}
6403   }
6404   }{
6405     \int_set:Nn \l_document_structure_section_level_int {2}
6406     \def\part@prefix{}
6407   }
6408 }
6409
6410 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `omgroup` environment that choses the L^AT_EX sectioning macros according to `\section@level`.

sfragment

```

6411 \renewenvironment{sfragment}[2][]{
6412   \_document_structure_omgroup_args:n { #1 }
6413   \int_incr:N \l_document_structure_section_level_int
6414   \bool_if:NT \c__notesslides_sectocframes_bool {
6415     \stepcounter{slide}
6416     \begin{frame}[noframenumbering]
6417       \vfill\Large\centering
6418       \red{
6419         \ifcase\l_document_structure_section_level_int\or
6420           \stepcounter{part}
6421           \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6422           \def\currentsectionlevel{\omdoc@part@kw}
6423         \or
6424           \stepcounter{chapter}
6425           \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6426           \def\currentsectionlevel{\omdoc@chapter@kw}
6427         \or
6428           \stepcounter{section}
6429           \def\_notesslideslabel{\part@prefix\arabic{section}}
6430           \def\currentsectionlevel{\omdoc@section@kw}
6431         \or
6432           \stepcounter{subsection}
6433           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6434           \def\currentsectionlevel{\omdoc@subsection@kw}
6435         \or
6436           \stepcounter{subsubsection}
6437           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6438           \def\currentsectionlevel{\omdoc@subsubsection@kw}
6439         \or
6440           \stepcounter{paragraph}
6441           \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6442           \def\currentsectionlevel{\omdoc@paragraph@kw}
6443         \else
6444           \def\_notesslideslabel{}

```

```

6445         \def\currentsectionlevel{\omdoc@paragraph@kw}
6446         \fi% end ifcase
6447         \_notesslideslabel%\sref@label@id\_notesslideslabel
6448         \quad #2%
6449     }%
6450     \vfill%
6451     \end{frame}%
6452 }
6453 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6454     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6455 }
6456 }{}
6457 }

```

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

```

6458 \def\inserttheorembodyfont{\normalfont}
6459 %\bool_if:NF \c__notesslides_notes_bool {
6460 % \defbeamertemplate{theorem begin}{miko}
6461 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6462 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6463 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6464 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

6455 % \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.

```

6466 % \expandafter\def\csname Parent2\endcsname{}
6467 %}
6468
6469 \AddToHook{begindocument}{% this does not work for some reason
6470     \setbeamertemplate{theorems}[ams style]
6471 }
6472 \bool_if:NT \c__notesslides_notes_bool {
6473     \renewenvironment{columns}[1][\]{%
6474         \par\noindent%
6475         \begin{minipage}%
6476             \slidewidth\centering\leavevmode%
6477     }{\%
6478         \end{minipage}\par\noindent%
6479     }%
6480     \newsavebox\columnbox%
6481     \renewenvironment<>{column}[2][\]{%
6482         \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6483     }{\%
6484         \end{minipage}\end{lrbox}\usebox\columnbox%
6485     }%
6486 }
6487 \bool_if:NTF \c__notesslides_noproblems_bool {
6488     \newenvironment{problems}{}{}
6489 }{
6490     \excludecomment{problems}
6491 }

```

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.

```

6492 \gdef\printexcursions{}
6493 \newcommand\excursionref[2]{% label, text
6494   \bool_if:NT \c__notesslides_notes_bool {
6495     \begin{sparagraph}[title=Excursion]
6496       #2 \sref[fallback=the appendix]{#1}.
6497     \end{sparagraph}
6498   }
6499 }
6500 \newcommand\activate@excursion[2][]{
6501   \gappto\printexcursions{\inputref{#1}{#2}}
6502 }
6503 \newcommand\excursion[4][]{% repos, label, path, text
6504   \bool_if:NT \c__notesslides_notes_bool {
6505     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6506   }
6507 }

```

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

`\excursiongroup`

```

6508 \keys_define:nn{notesslides / excursiongroup }{
6509   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6510   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6511   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6512 }
6513 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6514   \tl_clear:N \l__notesslides_excursion_intro_tl
6515   \str_clear:N \l__notesslides_excursion_id_str
6516   \str_clear:N \l__notesslides_excursion_mhrepos_str
6517   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6518 }
6519 \newcommand\excursiongroup[1][]{
6520   \__notesslides_excursion_args:n{ #1 }
6521   \ifdefempty\printexcursions{}% only if there are excursions
6522   {\begin{note}
6523     \begin{sfragment}[#1]{Excursions}%
6524     \ifdefempty\l__notesslides_excursion_intro_tl{\{
6525       \inputref[\l__notesslides_excursion_mhrepos_str]{
6526         \l__notesslides_excursion_intro_tl
6527       }
6528     }
6529     \printexcursions%
6530     \end{sfragment}
6531   }
6532 }
6533 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6534 \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.

```
6535 <*package>
6536 <@@=problems>
6537 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6538 \RequirePackage{l3keys2e,stex}
6539
6540 \keys_define:nn { problem / pkg }{
6541   notes      .default:n    = { true },
6542   notes      .bool_set:N   = \c__problems_notes_bool,
6543   gnotes     .default:n    = { true },
6544   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6545   hints      .default:n    = { true },
6546   hints      .bool_set:N   = \c__problems_hints_bool,
6547   solutions  .default:n    = { true },
6548   solutions  .bool_set:N   = \c__problems_solutions_bool,
6549   pts        .default:n    = { true },
6550   pts        .bool_set:N   = \c__problems_pts_bool,
6551   min        .default:n    = { true },
6552   min        .bool_set:N   = \c__problems_min_bool,
6553   boxed      .default:n    = { true },
6554   boxed      .bool_set:N   = \c__problems_boxed_bool,
6555   unknown    .code:n       = {}
6556 }
6557 \newif\ifsolutions
6558
6559 \ProcessKeysOptions{ problem / pkg }
6560 \bool_if:NTF \c__problems_solutions_bool {
6561   \solutionstrue
6562 }{
6563   \solutionsfalse
6564 }
```

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

```
6565 \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.

```
6566 \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.

```
6567 \def\prob@problem@kw{Problem}
6568 \def\prob@solution@kw{Solution}
6569 \def\prob@hint@kw{Hint}
6570 \def\prob@note@kw{Note}
6571 \def\prob@gnote@kw{Grading}
6572 \def\prob@pt@kw{pt}
6573 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6574 \AddToHook{begindocument}{
6575   \ltx@ifpackageloaded{babel}{
6576     \makeatletter
6577     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6578     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6579       \input{problem-ngerman.ldf}
6580     }
6581     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6582       \input{problem-finnish.ldf}
6583     }
6584     \clist_if_in:NnT \l_tmpa_clist {french}{
6585       \input{problem-french.ldf}
6586     }
6587     \clist_if_in:NnT \l_tmpa_clist {russian}{
6588       \input{problem-russian.ldf}
6589     }
6590     \makeatother
6591   }{ }
6592 }
```

40.2 Problems and Solutions

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

```
6593 \keys_define:nn{ problem / problem }{
6594   id      .str_set:x:N = \l__problems_prob_id_str,
6595   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6596   min     .tl_set:N    = \l__problems_prob_min_tl,
6597   title   .tl_set:N    = \l__problems_prob_title_tl,
6598   type    .tl_set:N    = \l__problems_prob_type_tl,
6599   refnum  .int_set:N   = \l__problems_prob_refnum_int
6600 }
6601 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6602 \str_clear:N \l__problems_prob_id_str
6603 \tl_clear:N \l__problems_prob_pts_tl
6604 \tl_clear:N \l__problems_prob_min_tl
6605 \tl_clear:N \l__problems_prob_title_tl
6606 \tl_clear:N \l__problems_prob_type_tl
6607 \int_zero_new:N \l__problems_prob_refnum_int
6608 \keys_set:nn { problem / problem }{ #1 }
6609 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6610   \let\l__problems_prob_refnum_int\undefined
6611 }
6612 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6613 \newcounter{problem}
6614 \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.

```

6615 \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

```

6616 \newcommand\prob@number{
6617   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6618     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6619   }{
6620     \int_if_exist:NTF \l__problems_prob_refnum_int {
6621       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6622     }{
6623       \prob@label\theproblem
6624     }
6625   }
6626 }

```

(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.

```

6627 \newcommand\prob@title[3]{%
6628   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6629     #2 \l__problems_inclprob_title_tl #3
6630   }{
6631     \tl_if_exist:NTF \l__problems_prob_title_tl {
6632       #2 \l__problems_prob_title_tl #3
6633     }{
6634       #1
6635     }
6636   }
6637 }

```

(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.

```

6638 \def\prob@heading{
6639   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6640   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
6641 }

```

(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

```

6642 \newenvironment{sproblem}[1][{}]{
6643   \__problems_prob_args:n{#1}%\sref@target%
6644   \@in@omtexttrue% we are in a statement (for inline definitions)
6645   \stepcounter{problem}\record@problem
6646   \def\current@section@level{\prob@problem@kw}
6647   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6648     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6649   }{
6650     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6651   }
6652   \str_if_exist:NTF \l__problems_inclprob_id_str {
6653     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6654   }{
6655     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6656   }
6657
6658
6659   \clist_set:No \l_tmpa_clist \sproblemtype
6660   \tl_clear:N \l_tmpa_tl
6661   \clist_map_inline:Nn \l_tmpa_clist {
6662     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6663       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6664     }
6665   }
6666   \tl_if_empty:NTF \l_tmpa_tl {
6667     \__problems_sproblem_start:
6668   }{
6669     \l_tmpa_tl
6670   }
6671   \stex_ref_new_doc_target:n \sproblemid
6672 }{
6673   \clist_set:No \l_tmpa_clist \sproblemtype
6674   \tl_clear:N \l_tmpa_tl
6675   \clist_map_inline:Nn \l_tmpa_clist {
6676     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6677       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6678     }
6679   }

```



```

6679 }
6680 \tl_if_empty:NTF \l_tmpa_tl {
6681   \__problems_sproblem_end:
6682 }{
6683   \l_tmpa_tl
6684 }
6685
6686
6687 \smallskip
6688 }
6689
6690
6691 \cs_new_protected:Nn \__problems_sproblem_start: {
6692   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6693 }
6694 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6695
6696 \newcommand\stexpatchproblem[3][] {
6697   \str_set:Nx \l_tmpa_str{ #1 }
6698   \str_if_empty:NTF \l_tmpa_str {
6699     \tl_set:Nn \__problems_sproblem_start: { #2 }
6700     \tl_set:Nn \__problems_sproblem_end: { #3 }
6701   }{
6702     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6703     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6704   }
6705 }
6706
6707
6708 \bool_if:NT \c__problems_boxed_bool {
6709   \surroundwithmdframed{problem}
6710 }

```

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

```

6711 \def\record@problem{
6712   \protected@write\@auxout{}
6713   {
6714     \string\@problem{\prob@number}
6715     {
6716       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6717         \l__problems_inclprob_pts_tl
6718       }{
6719         \l__problems_prob_pts_tl
6720       }
6721     }%
6722     {
6723       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6724         \l__problems_inclprob_min_tl
6725       }{
6726         \l__problems_prob_min_tl
6727       }
6728     }
6729   }
6730 }

```

(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).

```
6731 \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.

```
6732 \keys_define:nn { problem / solution }{
6733   id                .str_set_x:N = \l__problems_solution_id_str ,
6734   for               .tl_set:N    = \l__problems_solution_for_tl ,
6735   height            .dim_set:N   = \l__problems_solution_height_dim ,
6736   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6737   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6738   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
6739 }
6740 \cs_new_protected:Nn \__problems_solution_args:n {
6741   \str_clear:N \l__problems_solution_id_str
6742   \tl_clear:N \l__problems_solution_for_tl
6743   \tl_clear:N \l__problems_solution_srccite_tl
6744   \clist_clear:N \l__problems_solution_creators_clist
6745   \clist_clear:N \l__problems_solution_contributors_clist
6746   \dim_zero:N \l__problems_solution_height_dim
6747   \keys_set:nn { problem / solution }{ #1 }
6748 }
```

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

```
6749 \newcommand\@startsolution[1][{}]{
6750   \__problems_solution_args:n { #1 }
6751   \@in@omtexttrue% we are in a statement.
6752   \bool_if:NF \c__problems_boxed_bool { \hrule }
6753   \smallskip\noindent
6754   {\textbf\prob@solution@kw : \enspace}
6755   \begin{small}
6756   \def\current@section@level{\prob@solution@kw}
6757   \ignorespacesandpars
6758 }
```

\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.

```
6759 \newcommand\startsolutions{
6760   \specialcomment{solution}{\@startsolution}{
6761     \bool_if:NF \c__problems_boxed_bool {
6762       \hrule\medskip
6763     }
6764     \end{small}%
6765   }
6766   \bool_if:NT \c__problems_boxed_bool {
6767     \surroundwithmdframed{solution}
6768   }
6769 }
```

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

\stopsolutions

```
6770 \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.

```
6771 \ifsolutions
6772   \startsolutions
6773 \else
6774   \stopsolutions
6775 \fi
```

exnote

```
6786 \bool_if:NTF \c__problems_notes_bool {
6787   \newenvironment{exnote}[1][]{
6788     \par\smallskip\hrule\smallskip
6789     \noindent\textbf{\prob@note@kw : }\small
6790   }{
6791     \smallskip\hrule
6792   }
6793 }{
6794   \excludecomment{exnote}
6795 }
```

hint

```
6786 \bool_if:NTF \c__problems_notes_bool {
6787   \newenvironment{hint}[1][]{
6788     \par\smallskip\hrule\smallskip
6789     \noindent\textbf{\prob@hint@kw :~ }\small
6790   }{
6791     \smallskip\hrule
6792   }
6793   \newenvironment{exhint}[1][]{
6794     \par\smallskip\hrule\smallskip
6795     \noindent\textbf{\prob@hint@kw :~ }\small
6796   }{
6797     \smallskip\hrule
6798   }
6799 }{
6800   \excludecomment{hint}
6801   \excludecomment{exhint}
6802 }
```

gnote

```
6803 \bool_if:NTF \c__problems_notes_bool {
6804   \newenvironment{gnote}[1][]{
6805     \par\smallskip\hrule\smallskip
6806     \noindent\textbf{\prob@gnote@kw : }\small
6807   }{
6808     \smallskip\hrule
6809   }
6810 }{
6811   \excludecomment{gnote}
6812 }
```

40.3 Multiple Choice Blocks

EdN:22

mcb 22

```
6813 \newenvironment{mcb}{
6814   \begin{enumerate}
6815 }{
6816   \end{enumerate}
6817 }
```

we define the keys for the mcc macro

```
6818 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6819   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6820     \bool_set_true:N #1
6821   }{
6822     \bool_set_false:N #1
6823   }
6824 }
6825 \keys_define:nn { problem / mcc }{
6826   id          .str_set:N = \l__problems_mcc_id_str ,
6827   feedback    .tl_set:N  = \l__problems_mcc_feedback_tl ,
6828   T           .default:n  = { true } ,
6829   T           .bool_set:N = \l__problems_mcc_t_bool ,
6830   F           .default:n  = { true } ,
6831   F           .bool_set:N = \l__problems_mcc_f_bool ,
6832   Ttext       .code:n     = {
6833     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6834   } ,
6835   Ftext       .code:n     = {
6836     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6837   }
6838 }
6839 \cs_new_protected:Nn \l__problems_mcc_args:n {
6840   \str_clear:N \l__problems_mcc_id_str
6841   \tl_clear:N \l__problems_mcc_feedback_tl
6842   \bool_set_true:N \l__problems_mcc_t_bool
6843   \bool_set_true:N \l__problems_mcc_f_bool
6844   \bool_set_true:N \l__problems_mcc_Ttext_bool
6845   \bool_set_false:N \l__problems_mcc_Ftext_bool
6846   \keys_set:nn { problem / mcc }{ #1 }
6847 }
```

\mcc

```
6848 \newcommand\mcc[2][] {
6849   \l__problems_mcc_args:n{ #1 }
6850   \item #2
6851   \ifsolutions
6852     \
6853     \bool_if:NT \l__problems_mcc_t_bool {
6854       % TODO!
6855       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
6856     }
6857     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

6858      % TODO!
6859      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6860    }
6861    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6862      !
6863    }{
6864      \l__problems_mcc_feedback_tl
6865    }
6866    \fi
6867  } %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.

```

6868
6869 \keys_define:nn{ problem / inclproblem }{
6870   id      .str_set:N = \l__problems_inclprob_id_str,
6871   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6872   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6873   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6874   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6875   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6876   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6877 }
6878 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6879   \str_clear:N \l__problems_prob_id_str
6880   \tl_clear:N \l__problems_inclprob_pts_tl
6881   \tl_clear:N \l__problems_inclprob_min_tl
6882   \tl_clear:N \l__problems_inclprob_title_tl
6883   \tl_clear:N \l__problems_inclprob_type_tl
6884   \int_zero_new:N \l__problems_inclprob_refnum_int
6885   \str_clear:N \l__problems_inclprob_mhrepos_str
6886   \keys_set:nn { problem / inclproblem }{ #1 }
6887   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6888     \let\l__problems_inclprob_pts_tl\undefined
6889   }
6890   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6891     \let\l__problems_inclprob_min_tl\undefined
6892   }
6893   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6894     \let\l__problems_inclprob_title_tl\undefined
6895   }
6896   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6897     \let\l__problems_inclprob_type_tl\undefined
6898   }
6899   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6900     \let\l__problems_inclprob_refnum_int\undefined
6901   }
6902 }

```

```

6903
6904 \cs_new_protected:Nn \__problems_inclprob_clear: {
6905   \let\l__problems_inclprob_id_str\undefined
6906   \let\l__problems_inclprob_pts_tl\undefined
6907   \let\l__problems_inclprob_min_tl\undefined
6908   \let\l__problems_inclprob_title_tl\undefined
6909   \let\l__problems_inclprob_type_tl\undefined
6910   \let\l__problems_inclprob_refnum_int\undefined
6911   \let\l__problems_inclprob_mhrepos_str\undefined
6912 }
6913 \__problems_inclprob_clear:
6914
6915 \newcommand\includeproblem[2][ ]{
6916   \__problems_inclprob_args:n{ #1 }
6917   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6918     \input{#2}
6919   }{
6920     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6921       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6922     }
6923   }
6924   \__problems_inclprob_clear:
6925 }

```

(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.

```

6926 \AddToHook{enddocument}{
6927   \bool_if:NT \c__problems_pts_bool {
6928     \message{Total:~\arabic{pts}~points}
6929   }
6930   \bool_if:NT \c__problems_min_bool {
6931     \message{Total:~\arabic{min}~minutes}
6932   }
6933 }

```

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

```

6934 \def\pts#1{
6935   \bool_if:NT \c__problems_pts_bool {
6936     \marginpar{#1~\prob@pt@kw}
6937   }
6938 }
6939 \def\min#1{
6940   \bool_if:NT \c__problems_min_bool {
6941     \marginpar{#1~\prob@min@kw}
6942   }
6943 }

```

`\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.

```

6944 \newcounter{pts}
6945 \def\show@pts{
6946   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6947     \bool_if:NT \c__problems_pts_bool {
6948       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6949       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6950     }
6951   }{
6952     \tl_if_exist:NT \l__problems_prob_pts_tl {
6953       \bool_if:NT \c__problems_pts_bool {
6954         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6955         \addtocounter{pts}{\l__problems_prob_pts_tl}
6956       }
6957     }
6958   }
6959 }

```

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

`\show@min`

```

6960 \newcounter{min}
6961 \def\show@min{
6962   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6963     \bool_if:NT \c__problems_min_bool {
6964       \marginpar{\l__problems_inclprob_min_tl\ min}
6965       \addtocounter{min}{\l__problems_inclprob_min_tl}
6966     }
6967   }{
6968     \tl_if_exist:NT \l__problems_prob_min_tl {
6969       \bool_if:NT \c__problems_min_bool {
6970         \marginpar{\l__problems_prob_min_tl\ min}
6971         \addtocounter{min}{\l__problems_prob_min_tl}
6972       }
6973     }
6974   }
6975 }
6976 </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.

```
6977 <@@=hwexam>
6978 <*cls>
6979 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
6980 \RequirePackage{l3keys2e}
6981 \DeclareOption*{
6982   \PassOptionsToClass{\CurrentOption}{document-structure}
6983   \PassOptionsToPackage{\CurrentOption}{stex}
6984   \PassOptionsToPackage{\CurrentOption}{hwexam}
6985   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6986 }
6987 \ProcessOptions
```

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

```
6988 \LoadClass{document-structure}
6989 \RequirePackage{stex}
6990 \RequirePackage{hwexam}
6991 \RequirePackage{tikzinput}
6992 \RequirePackage{graphicx}
6993 \RequirePackage{a4wide}
6994 \RequirePackage{amssymb}
6995 \RequirePackage{amstext}
6996 \RequirePackage{amsmath}
```

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


```

6997 \newcommand\assig@default@type{\hwexam@assignment@kw}
6998 \def\document@hwexamtype{\assig@default@type}
6999 <@@=document_structure>
7000 \keys_define:nn { document-structure / document }{
7001 id .str_set_x:N = \c_document_structure_document_id_str,
7002 hwexamtype .tl_set:N = \document@hwexamtype
7003 }
7004 <@@=hwexam>
7005 </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.

```
7006 \*package>
7007 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7008 \RequirePackage{13keys2e}
7009
7010 \newif\iftest\testfalse
7011 \DeclareOption{test}{\testtrue}
7012 \newif\ifmultiple\multiplefalse
7013 \DeclareOption{multiple}{\multipletrue}
7014 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7015 \ProcessOptions
```

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

```
7016 \RequirePackage{keyval}[1997/11/10]
7017 \RequirePackage{problem}
```

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

```
7018 \newcommand\hwexam@assignment@kw{Assignment}
7019 \newcommand\hwexam@given@kw{Given}
7020 \newcommand\hwexam@due@kw{Due}
7021 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7022 blank~for~extra~space}
7023 \def\hwexam@minutes@kw{minutes}
7024 \newcommand\correction@probs@kw{prob.}
7025 \newcommand\correction@pts@kw{total}
7026 \newcommand\correction@reached@kw{reached}
7027 \newcommand\correction@sum@kw{Sum}
7028 \newcommand\correction@grade@kw{grade}
7029 \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

```

7030 \AddToHook{begindocument}{
7031 \ltx@ifpackageloaded{babel}{
7032 \makeatletter
7033 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7034 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7035 \input{hwexam-ngerman.ldf}
7036 }
7037 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7038 \input{hwexam-finnish.ldf}
7039 }
7040 \clist_if_in:NnT \l_tmpa_clist {french}{
7041 \input{hwexam-french.ldf}
7042 }
7043 \clist_if_in:NnT \l_tmpa_clist {russian}{
7044 \input{hwexam-russian.ldf}
7045 }
7046 \makeatother
7047 }{}
7048 }
7049

```

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.

```

7050 \newcounter{assignment}
7051 \numberproblemsin{assignment}
7052 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7053 \keys_define:nn { hwexam / assignment } {
7054 id .str_set:N = \l__hwexam_assign_id_str,
7055 number .int_set:N = \l__hwexam_assign_number_int,
7056 title .tl_set:N = \l__hwexam_assign_title_tl,
7057 type .tl_set:N = \l__hwexam_assign_type_tl,
7058 given .tl_set:N = \l__hwexam_assign_given_tl,
7059 due .tl_set:N = \l__hwexam_assign_due_tl,
7060 loadmodules .code:n = {
7061 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7062 }
7063 }
7064 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7065 \str_clear:N \l__hwexam_assign_id_str
7066 \int_set:Nn \l__hwexam_assign_number_int {-1}
7067 \tl_clear:N \l__hwexam_assign_title_tl
7068 \tl_clear:N \l__hwexam_assign_type_tl
7069 \tl_clear:N \l__hwexam_assign_given_tl
7070 \tl_clear:N \l__hwexam_assign_due_tl
7071 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7072 \keys_set:nn { hwexam / assignment }{ #1 }
7073 }

```

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.

```

7074 \newcommand\given@due[2]{
7075 \bool_lazy_all:nF {
7076 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7077 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7078 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7079 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7080 }{ #1 }
7081
7082 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7083 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7084 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7085 }
7086 }{
7087 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7088 }
7089
7090 \bool_lazy_or:nnF {
7091 \bool_lazy_and_p:nn {
7092 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7093 }{
7094 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7095 }
7096 }{
7097 \bool_lazy_and_p:nn {
7098 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7099 }{
7100 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7101 }
7102 }{ ,~ }
7103
7104 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7105 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7106 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7107 }
7108 }{
7109 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7110 }
7111
7112 \bool_lazy_all:nF {
7113 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7114 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7115 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7116 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7117 }{ #2 }
7118 }

```

`\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.

```

7119 \newcommand\assignment@title[3]{
7120 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
7121 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7122 #1
7123 }{
7124 #2\l__hwexam_assign_title_tl#3
7125 }
7126 }{
7127 #2\l__hwexam_inclassassign_title_tl#3
7128 }
7129 }

```

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

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

```

7130 \newcommand\assignment@number{
7131 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
7132 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7133 \arabic{assignment}
7134 } {
7135 \int_use:N \l__hwexam_assign_number_int
7136 }
7137 }{
7138 \int_use:N \l__hwexam_inclassassign_number_int
7139 }
7140 }

```

(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.

```

7141 \newenvironment{assignment}[1][]{
7142 \__hwexam_assignment_args:n { #1 }
7143 %\sref@target
7144 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7145 \global\stepcounter{assignment}
7146 }{
7147 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7148 }
7149 \setcounter{problem}{0}
7150 \def\current@section@level{\document@hwexamtype}
7151 %\sref@label@id{\document@hwexamtype \thesection}
7152 \begin{@assignment}
7153 }{
7154 \end{@assignment}
7155 }

```

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

```

7156 \def\ass@title{
7157 \protect\document@hwexamtype~\arabic{assignment}
7158 \assignment@title{}\{;\}{}\} -- \given@due{}\}
7159 }
7160 \ifmultiple
7161 \newenvironment{@assignment}{
7162 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7163 \begin{sfragment}[loadmodules]{\ass@title}
7164 }{
7165 \begin{sfragment}{\ass@title}
7166 }
7167 }{
7168 \end{sfragment}
7169 }

```

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

```

7170 \else
7171 \newenvironment{@assignment}{
7172 \begin{center}\bf
7173 \Large@title\strut\
7174 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}\}
7175 \large\given@due{--;\}{}\}
7176 \end{center}
7177 }{}
7178 \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.

```

7179 \keys_define:nn { hwexam / inclassignment } {
7180 %id .str_set_x:N = \l__hwexam_assign_id_str,
7181 number .int_set:N = \l__hwexam_inclassign_number_int,
7182 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7183 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7184 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7185 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7186 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7187 }
7188 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7189 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7190 \tl_clear:N \l__hwexam_inclassign_title_tl
7191 \tl_clear:N \l__hwexam_inclassign_type_tl
7192 \tl_clear:N \l__hwexam_inclassign_given_tl
7193 \tl_clear:N \l__hwexam_inclassign_due_tl
7194 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7195 \keys_set:nn { hwexam / inclassignment }{ #1 }
7196 }
7197 \__hwexam_inclassignment_args:n {}
7198
7199 \newcommand\inputassignment[2][{}]{

```

```

7200 \_hwexam_inclassnment_args:n { #1 }
7201 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7202 \input{#2}
7203 }{
7204 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7205 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7206 }
7207 }
7208 \_hwexam_inclassnment_args:n {}
7209 }
7210 \newcommand\includeassignment[2][]{
7211 \newpage
7212 \inputassignment[#1]{#2}
7213 }

```

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

42.4 Typesetting Exams

\quizheading

```

7214 \ExplSyntaxOff
7215 \newcommand\quizheading[1]{%
7216 \def\@tas{#1}%
7217 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7218 \ifx\@tas\@empty\else%
7219 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7220 \fi%
7221 }
7222 \ExplSyntaxOn

```

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

\testheading

```

7223
7224 \def\hwexamheader{\input{hwexam-default.header}}
7225
7226 \def\hwexamminutes{
7227 \tl_if_empty:NTF \testheading@duration {
7228 {\testheading@min}~\hwexam@minutes@kw
7229 }{
7230 \testheading@duration
7231 }
7232 }
7233
7234 \keys_define:nn { hwexam / testheading } {
7235 min .tl_set:N = \testheading@min,
7236 duration .tl_set:N = \testheading@duration,
7237 reqpts .tl_set:N = \testheading@reqpts,
7238 tools .tl_set:N = \testheading@tools
7239 }
7240 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7241 \tl_clear:N \testheading@min
7242 \tl_clear:N \testheading@duration

```

```

7243 \tl_clear:N \testheading@reqpts
7244 \tl_clear:N \testheading@tools
7245 \keys_set:nn { hwexam / testheading }{ #1 }
7246 }
7247 \newenvironment{testheading}[1][]{
7248   \__hwexam_testheading_args:n{ #1 }
7249   \newcount\check@time\check@time=\testheading@min
7250   \advance\check@time by -\theassignment@totalmin
7251   \newif\if@bonuspoints
7252   \tl_if_empty:NTF \testheading@reqpts {
7253     \@bonuspointsfalse
7254   }{
7255     \newcount\bonus@pts
7256     \bonus@pts=\theassignment@totalpts
7257     \advance\bonus@pts by -\testheading@reqpts
7258     \edef\bonus@pts{\the\bonus@pts}
7259     \@bonuspointstrue
7260   }
7261   \edef\check@time{\the\check@time}
7262 }
7263 \makeatletter\hwexamheader\makeatother
7264 }{
7265 \newpage
7266 }

```

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

\testspace

```

7267 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7268 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7269 \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.

```

7270 <@=problems>
7271 \renewcommand\@problem[3]{
7272   \stepcounter{assignment@probs}
7273   \def\__problemspts{#2}
7274   \ifx\__problemspts\@empty\else
7275     \addtocounter{assignment@totalpts}{#2}
7276   \fi
7277   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7278   \xdef\correction@probs{\correction@probs & #1}%
7279   \xdef\correction@pts{\correction@pts & #2}
7280   \xdef\correction@reached{\correction@reached &}

```



```

7281 }
7282 <@@=hwexam>

```

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

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

```

7283 \newcounter{assignment@probs}
7284 \newcounter{assignment@totalpts}
7285 \newcounter{assignment@totalmin}
7286 \def\correction@probs{\correction@probs@kw}
7287 \def\correction@pts{\correction@pts@kw}
7288 \def\correction@reached{\correction@reached@kw}
7289 \stepcounter{assignment@probs}
7290 \newcommand\correction@table{
7291 \resizebox{\textwidth}{!}{%
7292 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7293 &\multicolumn{\theassignment@probs}{c|}||%|
7294 {\footnotesize\correction@forgrading@kw} &\\ \hline
7295 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7296 \correction@pts & \theassignment@totalpts & \\ \hline
7297 \correction@reached & & \[.7cm]\hline
7298 \end{tabular}}
7299 </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}

```