

The sTeX3 Package *

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

2022-02-28

Abstract

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

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

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

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

*Version 3.0 (last revised 2022-02-28)

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	136
31.1	Symbol Invocations	136
31.2	Terms	141
31.3	Notation Components	148
31.4	Variables	150
32	STeX-Structural Features Implementation	152
32.1	Imports with modification	152
32.2	The feature environment	159
32.3	Features	160
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,
- RuSTeX to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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 Note, that the CTAN repository for L^AT_EX packages may contain outdated versions of the sTeX package, so make sure, that your TEXMF system variable is configured such that the packages available in the linked repository are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

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

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

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

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

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

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

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

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

2.2 A First 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/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

```
\begin{smodule}{assoctest}
\symdef{foo}[args=1]{\comp{a:}#1\comp{;b:}#2\comp{;c:}#3}{\comp{#1\comp{;}#1\comp{##2\comp{;#2\comp{}}}
\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}
\symdef{universe}{\comp{\mathcal U}}
\symdef{operation}[args=2,op=\circ]{#1\comp{\circ}#2}
\end{smodule}
\begin{smodule}{monoid}
\importmodule{magma}
\symdef{unit}{\comp{e}}
\end{smodule}
\begin{smodule}{group}
\importmodule{monoid}
\symdef{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  $\textcolor{blue}{a}+\textcolor{blue}{b}$ .
```

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

`*` 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. `\stex` determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

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

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

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

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

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

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

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

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

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

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

Chapter 14

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 `<macroname>`.
- **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\Nat}{x\geq0}`.

<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>s (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>s (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>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <hr/>	
<code>\STEXinvisible</code>	Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <hr/>	
<code>\ellipses</code>	TODO

Chapter 16

TeX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

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}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

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

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

18.2.4 Proof Structure

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

18.2.5 Proof End Markers

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

\sproofend	The <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i>), then this can be obtained by specifying it using the
\sProofEndSymbol	<code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code>).
Some of the proof structuring macros above will insert proof end symbols for subproofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> .	

18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

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

18.3 Limitations

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

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

Chapter 19

sTeX-Metatheory

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

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

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

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

Chapter 21

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

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

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

21.1 Introduction

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

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

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

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

21.2 The User Interface

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

21.2.1 Package and Class Options

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

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

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

21.2.2 Document Structure

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

```

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

```

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

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

`blindfragment`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

22.2.4 Frame Images

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

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

```
\begin{note}
\begin{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=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants 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=elefants,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 Elefants)
How many Elefants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Name: _____ Matriculation Number: _____

2022-02-28

Write the solutions to the sheet.

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

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

[illegible]

Example 8: A generated test heading.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%%% basics.dtx %%%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/02/28}{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}{2022/02/28}{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 `LATXML`:

`\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_ampersand_str lrm; }
93   }{-}
94 }

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

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

```

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

```

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

```

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

```



```

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 }
1075
\l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
(End definition for \l_stex_current_module_str. This variable is documented on page 28.)

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

```

```

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
    Warnings and error messages
1818 \msg_new:nnn{stex}{error/wrongargs}{
1819   args~value~in~symbol~declaration~for~#1~
1820   needs~to~be~i,~a,~b~or~B,~but~#2~given
1821 }
```

30.1 Symbol Declarations

```
1822 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols

1823 \cs_new_protected:Nn \stex_all_symbols:n {
1824   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1825   \seq_map_inline:Nn \l_stex_all_modules_seq {
1826     \seq_map_inline:cn{c_stex_module_##1_constants}{
1827       \__stex_symdecl_all_symbols_cs{##1?###1}
1828     }
1829   }
1830 }

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

\STEXsymbol

1831 \NewDocumentCommand \STEXsymbol { m } {
1832   \stex_get_symbol:n { #1 }
1833   \exp_args:No
1834   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1835 }
```

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

`symdecl` arguments:

```

1836 \keys_define:nn { stex / symdecl } {
1837   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1838   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1839   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1840   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1841   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1842   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1843   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1844   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1845   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1846   assoc     .choices:nn =
1847     {bin,binl,binr,pre,conj,pwconj}
1848     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
1849 }
1850
1851 \bool_new:N \l_stex_symdecl_make_macro_bool
1852
1853 \cs_new_protected:Nn \__stex_symdecl_args:n {
1854   \str_clear:N \l_stex_symdecl_name_str
1855   \str_clear:N \l_stex_symdecl_args_str
1856   \str_clear:N \l_stex_symdecl_deprecate_str
1857   \str_clear:N \l_stex_symdecl_assoctype_str
1858   \bool_set_false:N \l_stex_symdecl_local_bool
1859   \tl_clear:N \l_stex_symdecl_type_tl
1860   \tl_clear:N \l_stex_symdecl_definiens_tl
1861
1862   \keys_set:nn { stex / symdecl } { #1 }
1863 }

```

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

```

1864
1865 \NewDocumentCommand \symdecl { s m O{} } {
1866   \__stex_symdecl_args:n { #3 }
1867   \IfBooleanTF #1 {
1868     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1869   } {
1870     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1871   }
1872   \stex_symdecl_do:n { #2 }
1873   \stex_smsmode_do:
1874 }
1875
1876 \cs_new_protected:Nn \stex_symdecl_do:nn {
1877   \__stex_symdecl_args:n{#1}
1878   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1879   \stex_symdecl_do:n{#2}
1880 }
1881
1882 \stex_deactivate_macro:Nn \symdecl {module~environments}

```

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

`\stex_symdecl_do:n`

```
1883 \cs_new_protected:Nn \stex_symdecl_do:n {
1884   \stex_if_in_module:F {
1885     % TODO throw error? some default namespace?
1886   }
1887
1888   \str_if_empty:NT \l_stex_symdecl_name_str {
1889     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1890   }
1891
1892   \prop_if_exist:cT { l_stex_symdecl_
1893     \l_stex_current_module_str ?
1894     \l_stex_symdecl_name_str
1895   }_prop
1896   }{
1897     % TODO throw error (beware of circular dependencies)
1898   }
1899
1900   \prop_clear:N \l_tmpa_prop
1901   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1902   \seq_clear:N \l_tmpa_seq
1903   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1904   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1905
1906   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1907     \str_if_empty:NF \l_stex_module_deprecate_str {
1908       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1909     }
1910   }
1911   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1912
1913   \exp_args:No \stex_add_constant_to_current_module:n {
1914     \l_stex_symdecl_name_str
1915   }
1916
1917   % arity/args
1918   \int_zero:N \l_tmpb_int
1919
1920   \bool_set_true:N \l_tmpa_bool
1921   \str_map_inline:Nn \l_stex_symdecl_args_str {
1922     \token_case_meaning:NnF ##1 {
1923       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1924       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1925       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1926       {\tl_to_str:n a} {
1927         \bool_set_false:N \l_tmpa_bool
1928         \int_incr:N \l_tmpb_int
1929       }
1930       {\tl_to_str:n B} {
1931         \bool_set_false:N \l_tmpa_bool
1932         \int_incr:N \l_tmpb_int
1933       }
1934     }{
1935       \msg_error:nnxx{stex}{error/wrongargs}{
```

```

1936         \l_stex_current_module_str ?
1937         \l_stex_symdecl_name_str
1938     }{##1}
1939 }
1940 }
1941 \bool_if:NTF \l_tmpa_bool {
1942     % possibly numeric
1943     \str_if_empty:NTF \l_stex_symdecl_args_str {
1944         \prop_put:Nnn \l_tmpa_prop { args } {}
1945         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1946     }{
1947         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1948         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1949         \str_clear:N \l_tmpa_str
1950         \int_step_inline:nn \l_tmpa_int {
1951             \str_put_right:Nn \l_tmpa_str i
1952         }
1953         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1954     }
1955 } {
1956     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1957     \prop_put:Nnx \l_tmpa_prop { arity }
1958     { \str_count:N \l_stex_symdecl_args_str }
1959 }
1960 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1961
1962
1963 % semantic macro
1964
1965 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1966     \exp_args:Nx \stex_do_up_to_module:n {
1967         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1968             \l_stex_current_module_str ? \l_stex_symdecl_name_str
1969         }}
1970     }
1971
1972     \bool_if:NF \l_stex_symdecl_local_bool {
1973         \exp_args:Nx \stex_add_to_current_module:n {
1974             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1975                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
1976             } }
1977         }
1978     }
1979 }
1980
1981 \stex_debug:nn{symbols}{New~symbol:~
1982     \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1983     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1984     Args:~\prop_item:Nn \l_tmpa_prop { args }
1985 }
1986
1987 % circular dependencies require this:
1988
1989 \prop_if_exist:cF {

```

```

1990     \l_stex_symdecl_
1991     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1992     _prop
1993   } {
1994     \prop_set_eq:cN {
1995       \l_stex_symdecl_
1996       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1997       _prop
1998     } \l_tmpa_prop
1999   }
2000
2001   \seq_clear:c {
2002     \l_stex_symdecl_
2003     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2004     _notations
2005   }
2006
2007   \bool_if:NF \l_stex_symdecl_local_bool {
2008     \exp_args:Nx
2009     \stex_add_to_current_module:n {
2010       \seq_clear:c {
2011         \l_stex_symdecl_
2012         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013         _notations
2014       }
2015       \prop_set_from_keyval:cn {
2016         \l_stex_symdecl_
2017         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018         _prop
2019       } {
2020         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2021         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2022         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2023         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2024         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2025         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2026       }
2027     }
2028   }
2029
2030   \stex_if_smsmode:F {
2031     % \exp_args:Nx \stex_do_up_to_module:n {
2032     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2033     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2034     %   }
2035     % }
2036   \stex_if_do_html:T {
2037     \stex_annotate_invisible:nnn {symdecl} {
2038       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2039     } {
2040       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2041       \stex_annotate_invisible:nnn{args}{}}{
2042         \prop_item:Nn \l_tmpa_prop { args }
2043       }

```

```

2044 \stex_annotate_invisible:nnn{macroname}{#1}{ }
2045 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2046   \stex_annotate_invisible:nnn{definiens}{ }
2047   { $\l_stex_symdecl_definiens_tl$ }
2048 }
2049 \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2050   \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{ }
2051 }
2052 }
2053 }
2054 }
2055 }

```

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

`\stex_get_symbol:n`

```

2056 \str_new:N \l_stex_get_symbol_uri_str
2057
2058 \cs_new_protected:Nn \stex_get_symbol:n {
2059   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2060     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2061   }{
2062     % argument is a string
2063     % is it a command name?
2064     \cs_if_exist:cTF { #1 }{
2065       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2066       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2067       \str_if_empty:NTF \l_tmpa_str {
2068         \exp_args:Nx \cs_if_eq:NNTF {
2069           \tl_head:N \l_tmpa_tl
2070         } \stex_invoke_symbol:n {
2071           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2072         }{
2073           \__stex_symdecl_get_symbol_from_string:n { #1 }
2074         }
2075       } {
2076         \__stex_symdecl_get_symbol_from_string:n { #1 }
2077       }
2078     }{
2079       % argument is not a command name
2080       \__stex_symdecl_get_symbol_from_string:n { #1 }
2081       % \l_stex_all_symbols_seq
2082     }
2083   }
2084   \str_if_eq:eeF {
2085     \prop_item:cn {
2086       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2087     }{ deprecate }
2088   }{}{
2089     \msg_warning:nxxx{stex}{warning/deprecated}{
2090       Symbol~\l_stex_get_symbol_uri_str
2091     }{
2092       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2093     }
2094   }

```

```

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

```

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

30.2 Notations

```

2136 <@=stex_notation>
      notation arguments:
2137 \keys_define:nn { stex / notation } {
2138   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2139   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2140   prec      .str_set_x:N = \l__stex_notation_prec_str ,

```

```

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

```

\notation

```

2157 \NewDocumentCommand \notation { s m O{} } {
2158 \stex_notation_args:n { #3 }
2159 \tl_clear:N \l_stex_symdecl_definiens_tl
2160 \stex_get_symbol:n { #2 }
2161 \tl_set:Nn \l_stex_notation_after_do_tl {
2162 \__stex_notation_final:
2163 \IfBooleanTF#1{
2164 \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2165 }{}
2166 \stex_smsmode_do:
2167 }
2168 \stex_notation_do:nnnn
2169 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { args } }
2170 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { arity } }
2171 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2172 }
2173 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnn

```

2174 \seq_new:N \l__stex_notation_precedences_seq
2175 \tl_new:N \l__stex_notation_opprec_tl
2176 \int_new:N \l__stex_notation_currarg_int
2177 \tl_new:N \stex_symbol_after_invokation_tl
2178
2179 \cs_new_protected:Nn \stex_notation_do:nnnn {
2180 \let\l_stex_current_symbol_str\relax
2181 \seq_clear:N \l__stex_notation_precedences_seq
2182 \tl_clear:N \l__stex_notation_opprec_tl
2183 \str_set:Nx \l__stex_notation_args_str { #1 }
2184 \str_set:Nx \l__stex_notation_arity_str { #2 }
2185 \str_set:Nx \__stex_notation_suffix_str { #3 }
2186
2187 % precedences
2188 \str_if_empty:NTF \l__stex_notation_prec_str {
2189 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {

```



```

2190     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2191   }{
2192     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2193   }
2194 } {
2195   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2196     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2197     \int_step_inline:nn { \l__stex_notation_arity_str } {
2198       \exp_args:NNo
2199       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2200     }
2201   }{
2202     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2203     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2204       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2205       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2206         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2207           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2208         \seq_map_inline:Nn \l_tmpa_seq {
2209           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2210         }
2211       }
2212     }{
2213       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2214         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2215       }{
2216         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2217       }
2218     }
2219   }
2220 }
2221
2222 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2223 \int_step_inline:nn { \l__stex_notation_arity_str } {
2224   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2225     \exp_args:NNo
2226     \seq_put_right:No \l__stex_notation_precedences_seq {
2227       \l__stex_notation_opprec_tl
2228     }
2229   }
2230 }
2231 \tl_clear:N \l_stex_notation_dummyargs_tl
2232
2233 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2234   \exp_args:NNe
2235   \cs_set:Npn \l_stex_notation_macrocode_cs {
2236     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2237     { \_stex_notation_suffix_str }
2238     { \l__stex_notation_opprec_tl }
2239     { \exp_not:n { #4 } }
2240   }
2241   \l_stex_notation_after_do_tl
2242 }{
2243   \str_if_in:NnTF \l__stex_notation_args_str b {

```

```

2244 \exp_args:Nne \use:nn
2245 {
2246 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2247 \cs_set:Npn \l__stex_notation_arity_str } { {
2248 \stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2249 { \__stex_notation_suffix_str }
2250 { \l__stex_notation_opprec_tl }
2251 { \exp_not:n { #4 } }
2252 }}
2253 }{
2254 \str_if_in:NnTF \l__stex_notation_args_str B {
2255 \exp_args:Nne \use:nn
2256 {
2257 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2258 \cs_set:Npn \l__stex_notation_arity_str } { {
2259 \stex_term_math_omb:nmmm { \l_stex_current_symbol_str }
2260 { \__stex_notation_suffix_str }
2261 { \l__stex_notation_opprec_tl }
2262 { \exp_not:n { #4 } }
2263 } }
2264 }{
2265 \exp_args:Nne \use:nn
2266 {
2267 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2268 \cs_set:Npn \l__stex_notation_arity_str } { {
2269 \stex_term_math_oma:nmmm { \l_stex_current_symbol_str }
2270 { \__stex_notation_suffix_str }
2271 { \l__stex_notation_opprec_tl }
2272 { \exp_not:n { #4 } }
2273 } }
2274 }
2275 }
2276
2277 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2278 \int_zero:N \l__stex_notation_currarg_int
2279 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2280 \__stex_notation_arguments:
2281 }
2282 }

```

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

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

```

2283 \cs_new_protected:Nn \__stex_notation_arguments: {
2284 \int_incr:N \l__stex_notation_currarg_int
2285 \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2286 \l_stex_notation_after_do_tl
2287 }{
2288 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2289 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2290 \str_if_eq:NnTF \l_tmpa_str a {
2291 \__stex_notation_argument_assoc:n
2292 }{
2293 \str_if_eq:NnTF \l_tmpa_str B {

```

```

2294     \_stex_notation_argument_assoc:n
2295   }{
2296     \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2297     \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2298       { \_stex_term_math_arg:nnn
2299         { \int_use:N \l__stex_notation_currarg_int }
2300         { \l_tmpa_str }
2301         { ####\int_use:N \l__stex_notation_currarg_int }
2302       }
2303     }
2304     \_stex_notation_arguments:
2305   }
2306 }
2307 }
2308 }

```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2309 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2310
2311   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2312     {\l__stex_notation_arity_str}{
2313       #1
2314     }
2315   \int_zero:N \l_tmpa_int
2316   \tl_clear:N \l_tmpa_tl
2317   \str_map_inline:Nn \l__stex_notation_args_str {
2318     \int_incr:N \l_tmpa_int
2319     \tl_put_right:Nx \l_tmpa_tl {
2320       \str_if_eq:nnTF {##1}{a}{ {} } {}
2321       \str_if_eq:nnTF {##1}{B}{ {} } {}
2322       {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2323       }
2324     }
2325   }
2326 }
2327 \exp_after:wN\exp_after:wN\exp_after:wN \def
2328 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2329 \exp_after:wN\exp_after:wN\exp_after:wN ##
2330 \exp_after:wN\exp_after:wN\exp_after:wN 1
2331 \exp_after:wN\exp_after:wN\exp_after:wN ##
2332 \exp_after:wN\exp_after:wN\exp_after:wN 2
2333 \exp_after:wN\exp_after:wN\exp_after:wN {
2334   \exp_after:wN \exp_after:wN \exp_after:wN
2335   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2336     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2337   }
2338 }
2339
2340 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2341 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2342   \_stex_term_math_assoc_arg:nnnn
2343   { \int_use:N \l__stex_notation_currarg_int }

```

```

2344     { \l_tmpa_str }
2345     { ####\int_use:N \l__stex_notation_currarg_int }
2346     { \l_tmpa_cs {####1} {####2} }
2347   } }
2348   \__stex_notation_arguments:
2349 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2350 \cs_new_protected:Nn \__stex_notation_final: {
2351   \exp_args:Nne \use:nn
2352   {
2353     \cs_generate_from_arg_count:cNnn {
2354       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2355       \__stex_notation_suffix_str
2356       _cs
2357     }
2358     \cs_set:Npn \l__stex_notation_arity_str { { {
2359       \exp_after:wN \exp_after:wN \exp_after:wN
2360       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2361       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2362     } } }
2363
2364     \tl_if_empty:NF \l__stex_notation_op_tl {
2365       \cs_set:cpx {
2366         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2367         \__stex_notation_suffix_str
2368         _cs
2369       } {
2370         \stex_term oms:nnn {
2371           \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2372         }{
2373           \l_stex_get_symbol_uri_str
2374         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2375       }
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         \__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           \__stex_notation_suffix_str
2393           _cs

```

```

2394     } {
2395         \stex_term_oms:nnn {
2396             \l_stex_get_symbol_uri_str\c_hash_str \_stex_notation_suffix_str
2397         }{
2398             \l_stex_get_symbol_uri_str
2399         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2400     }
2401 }
2402 }
2403 %\exp_args:Nx
2404 % \stex_do_up_to_module:n {
2405     \seq_put_right:cx {
2406         l_stex_symdecl_ \l_stex_get_symbol_uri_str
2407         _notations
2408     } {
2409         \_stex_notation_suffix_str
2410     }
2411 % }
2412
2413 \stex_debug:nn{symbols}{
2414     Notation~\_stex_notation_suffix_str
2415     ~for~\l_stex_get_symbol_uri_str^^J
2416     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2417     Argument~precedences:~
2418     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2419     Notation: \cs_meaning:c {
2420         stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2421         \_stex_notation_suffix_str
2422         _cs
2423     }
2424 }
2425
2426 \exp_args:Ne
2427 \stex_add_to_current_module:n {
2428     \seq_put_right:cn {
2429         l_stex_symdecl_\l_stex_get_symbol_uri_str
2430         _notations
2431     } { \_stex_notation_suffix_str }
2432 }
2433
2434 \stex_if_smsmode:F {
2435
2436     % HTML annotations
2437     \stex_if_do_html:T {
2438         \stex_annotate_invisible:nnn { notation }
2439         { \l_stex_get_symbol_uri_str } {
2440             \stex_annotate_invisible:nnn { notationfragment }
2441             { \_stex_notation_suffix_str }{}
2442             \stex_annotate_invisible:nnn { precedence }
2443             { \l__stex_notation_prec_str }{}
2444
2445             \int_zero:N \l_tmpa_int
2446             \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2447             \tl_clear:N \l_tmpa_tl

```

```

2448 \int_step_inline:nn { \l__stex_notation_arity_str }{
2449 \int_incr:N \l_tmpa_int
2450 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2451 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2452 \str_if_eq:VnTF \l_tmpb_str a {
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 \str_if_eq:VnTF \l_tmpb_str B {
2459 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2460 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2461 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2462 } }
2463 }{
2464 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2465 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2466 } }
2467 }
2468 }
2469 }
2470 \stex_annotate_invisible:nnn { notationcomp }{}{
2471 \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2472 $ \exp_args:Nno \use:nn { \use:c {
2473 stex_notation_ \l_stex_current_symbol_str
2474 \c_hash_str \__stex_notation_suffix_str _cs
2475 } } { \l_tmpa_tl } $
2476 }
2477 }
2478 }
2479 }
2480 }

```

(End definition for __stex_notation_final:.)

\setnotation

```

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

```

```

2498     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2499 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2500     { \c_hash_str }
2501 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2502     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2503 \exp_args:Nx \stex_add_to_current_module:n {
2504     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2505     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2506     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2507     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2508     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2509     { \c_hash_str }
2510 }
2511 \stex_debug:nn {notations}{
2512     Setting~default~notation~
2513     {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2514     #1 \\\
2515     \expandafter\meaning\csname
2516     l_stex_symdecl_#1 _notations\endcsname
2517 }
2518 }{
2519     % todo throw error
2520 }
2521 }
2522
2523 \NewDocumentCommand \setnotation {m m} {
2524     \stex_get_symbol:n { #1 }
2525     \_stex_setnotation_args:n { #2 }
2526     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2527     \stex_smsmode_do:
2528 }
2529
2530 \cs_new_protected:Nn \stex_copy_notations:nn {
2531     \stex_debug:nn {notations}{
2532         Copying~notations~from~#2~to~#1\\
2533         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2534     }
2535     \tl_clear:N \l_tmpa_tl
2536     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2537         \tl_put_right:Nn \l_tmpa_tl { {##} ##1 }
2538     }
2539     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2540         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2541         \edef \l_tmpa_tl {
2542             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2543             \exp_after:wN\exp_after:wN\exp_after:wN {
2544                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2545             }
2546         }
2547         \exp_args:Nx
2548         \stex_do_up_to_module:n {
2549             \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2550             \cs_generate_from_arg_count:cNnn {
2551                 stex_notation_ #1 \c_hash_str ##1 _cs

```

```

2552     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2553       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2554     }
2555   }
2556 }
2557 }
2558
2559 \NewDocumentCommand \copynotation {m m} {
2560   \stex_get_symbol:n { #1 }
2561   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2562   \stex_get_symbol:n { #2 }
2563   \exp_args:Noo
2564   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2565   \exp_args:Nx \stex_add_import_to_current_module:n{
2566     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2567   }
2568   \stex_smsmode_do:
2569 }
2570

```

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

\symdef

```

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

```



```

2602
2603 \NewDocumentCommand \symdef { m O{} } {
2604   \__stex_notation_symdef_args:n { #2 }
2605   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2606   \stex_symdecl_do:n { #1 }
2607   \tl_set:Nn \l_stex_notation_after_do_tl {
2608     \__stex_notation_final:
2609     \stex_smsmode_do:
2610   }
2611   \str_set:Nx \l_stex_get_symbol_uri_str {
2612     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2613   }
2614   \exp_args:Nx \stex_notation_do:nnnn
2615     { \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str_prop } { args } }
2616     { \prop_item:cn { l_stex_symdecl_l_stex_get_symbol_uri_str_prop } { arity } }
2617     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2618   }
2619   \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page [35](#).)

30.3 Variables

```

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

```

```

2651 \__stex_variables_args:n {#2}
2652 \str_if_empty:NT \l__stex_variables_name_str {
2653   \str_set:Nx \l__stex_variables_name_str { #1 }
2654 }
2655 \prop_clear:N \l_tmpa_prop
2656 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2657
2658 \int_zero:N \l_tmpb_int
2659 \bool_set_true:N \l_tmpa_bool
2660 \str_map_inline:Nn \l__stex_variables_args_str {
2661   \token_case_meaning:NnF ##1 {
2662     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2663     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2664     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2665     {\tl_to_str:n a} {
2666       \bool_set_false:N \l_tmpa_bool
2667       \int_incr:N \l_tmpb_int
2668     }
2669     {\tl_to_str:n B} {
2670       \bool_set_false:N \l_tmpa_bool
2671       \int_incr:N \l_tmpb_int
2672     }
2673   }{
2674     \msg_error:nnxx{stex}{error/wrongargs}{
2675       variable~\l__stex_variables_name_str
2676     }{##1}
2677   }
2678 }
2679 \bool_if:NTF \l_tmpa_bool {
2680   % possibly numeric
2681   \str_if_empty:NTF \l__stex_variables_args_str {
2682     \prop_put:Nnn \l_tmpa_prop { args } {}
2683     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2684   }{
2685     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2686     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2687     \str_clear:N \l_tmpa_str
2688     \int_step_inline:nn \l_tmpa_int {
2689       \str_put_right:Nn \l_tmpa_str i
2690     }
2691     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2692     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2693   }
2694 } {
2695   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2696   \prop_put:Nnx \l_tmpa_prop { arity }
2697   { \str_count:N \l__stex_variables_args_str }
2698 }
2699 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2700 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2701
2702 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2703
2704 \tl_if_empty:NF \l__stex_variables_op_tl {

```

```

2705 \cs_set:cpx {
2706   stex_var_op_notation_ \l__stex_variables_name_str _cs
2707 } {
2708   \stex_term_omv:nn {
2709     var://\l__stex_variables_name_str
2710   }{ \comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2711 }
2712 }
2713
2714 \tl_set:Nn \l_stex_notation_after_do_tl {
2715   \exp_args:Nne \use:nn {
2716     \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
2717     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2718   } {
2719     \exp_after:wN \exp_after:wN \exp_after:wN
2720     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2721     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2722   }}
2723 \stex_if_do_html:T {
2724   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2725     \stex_annotate_invisible:nnn { precedence }
2726     { \l__stex_variables_prec_str }{
2727     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{
2728     \stex_annotate_invisible:nnn{args}{}{ \l__stex_variables_args_str }
2729     \stex_annotate_invisible:nnn{macroname}{#1}{
2730     \tl_if_empty:NF \l__stex_variables_def_tl {
2731       \stex_annotate_invisible:nnn{definiens}{
2732         { $\l__stex_variables_def_tl$ }
2733       }
2734     \str_if_empty:NF \l__stex_variables_assoctype_str {
2735       \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2736     }
2737     \int_zero:N \l_tmpa_int
2738     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2739     \tl_clear:N \l_tmpa_tl
2740     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2741       \int_incr:N \l_tmpa_int
2742       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2743       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2744     \str_if_eq:VnTF \l_tmpb_str a {
2745       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2746         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2747         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2748       } }
2749     }{
2750       \str_if_eq:VnTF \l_tmpb_str B {
2751         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2752           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2753           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2754         } }
2755       }{
2756         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2757           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2758         } }

```

```

2759         }
2760     }
2761 }
2762 \stex_annotate_invisible:nnn { notationcomp }{}{
2763     \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2764     $ \exp_args:Nno \use:nn { \use:c {
2765         stex_var_notation_\l__stex_variables_name_str _cs
2766     } } { \l_tmpa_tl } $
2767 }
2768 }
2769 }
2770 }
2771
2772 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { arit
2773 }
2774
2775 \cs_new:Nn \__stex_variables_reset:N {
2776     \tl_if_exist:NTF #1 {
2777         \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2778     }{
2779         \let \exp_not:N #1 \exp_not:N \undefined
2780     }
2781 }
2782
2783 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2784     \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2785     \exp_args:Nnx \use:nn {
2786         % TODO
2787         \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2788             #2
2789         }
2790     }{
2791         \__stex_variables_reset:N \varnot
2792         \__stex_variables_reset:N \vartype
2793         \__stex_variables_reset:N \vardefi
2794     }
2795 }
2796
2797 \NewDocumentCommand \vardef { s } {
2798     \IfBooleanTF#1 {
2799         \__stex_variables_do_complex:nn
2800     }{
2801         \__stex_variables_do_simple:nnn
2802     }
2803 }
2804
2805 \NewDocumentCommand \svar { 0{ } m }{
2806     \tl_if_empty:nTF {#1}{
2807         \str_set:Nn \l_tmpa_str { #2 }
2808     }{
2809         \str_set:Nn \l_tmpa_str { #1 }
2810     }
2811     \stex_term_omv:nn {
2812         var://\l_tmpa_str

```

```
2813     }{ \comp{ #2 } }  
2814 }  
2815  
2816 </package>
```

Chapter 31

STEX -Terms Implementation

```
2817 <*package>
2818
2819 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2820
2821 <@@=stex_terms>
2822
2823   Warnings and error messages
2824   \msg_new:nnn{stex}{error/nonotation}{
2825     Symbol~#1~invoked,~but~has~no~notation~#2!
2826   }
2827   \msg_new:nnn{stex}{error/notationarg}{
2828     Error~in~parsing~notation~#1
2829   }
2830   \msg_new:nnn{stex}{error/noop}{
2831     Symbol~#1~has~no~operator~notation~for~notation~#2
2832   }
2833   \msg_new:nnn{stex}{error/notallowed}{
2834     Symbol~invokation~#1~not~allowed~in~notation~component~of~#2
2835   }
```

31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2835 \keys_define:nn { stex / terms } {
2836   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2837   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2838   unknown .code:n = \str_set:Nx
2839     \l__stex_terms_variant_str \l_keys_key_str
2840 }
2841
2842 \cs_new_protected:Nn \__stex_terms_args:n {
2843   \str_clear:N \l__stex_terms_lang_str
2844   \str_clear:N \l__stex_terms_variant_str
2845 }
```

```

2846 \keys_set:nn { stex / terms } { #1 }
2847 }
2848
2849 \cs_new:Nn \__stex_terms_reset:N {
2850 \tl_if_exist:NTF #1 {
2851 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2852 }{
2853 \let \exp_not:N #1 \exp_not:N \undefined
2854 }
2855 }
2856
2857 \bool_new:N \l_stex_allow_semantic_bool
2858 \bool_set_true:N \l_stex_allow_semantic_bool
2859
2860 \cs_new_protected:Nn \stex_invoke_symbol:n {
2861 \bool_if:NTF \l_stex_allow_semantic_bool {
2862 \str_if_eq:eeF {
2863 \prop_item:cn {
2864 l_stex_symdecl_#1_prop
2865 }{ deprecate }
2866 }{}{
2867 \msg_warning:nxxx{stex}{warning/deprecated}{
2868 Symbol~#1
2869 }{
2870 \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2871 }
2872 }
2873 \if_mode_math:
2874 \exp_after:wN \__stex_terms_invoke_math:n
2875 \else:
2876 \exp_after:wN \__stex_terms_invoke_text:n
2877 \fi: { #1 }
2878 }{
2879 \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2880 }
2881 }
2882
2883 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2884 \peek_charcode_remove:NTF ! {
2885 \__stex_terms_invoke_op_custom:nn {#1}
2886 }{
2887 \__stex_terms_invoke_custom:nn {#1}
2888 }
2889 }
2890
2891 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2892 \peek_charcode_remove:NTF ! {
2893 % operator
2894 \peek_charcode_remove:NTF * {
2895 % custom op
2896 \__stex_terms_invoke_op_custom:nn {#1}
2897 }{
2898 % op notation
2899 \peek_charcode:NTF [ {

```

```

2900     \__stex_terms_invoke_op_notation:nw {#1}
2901   }{
2902     \__stex_terms_invoke_op_notation:nw {#1}[]
2903   }
2904 }
2905 }{
2906   \peek_charcode_remove:NTF * {
2907     \__stex_terms_invoke_custom:nn {#1}
2908     % custom
2909   }{
2910     % normal
2911     \peek_charcode:NTF [ {
2912       \__stex_terms_invoke_notation:nw {#1}
2913     }{
2914       \__stex_terms_invoke_notation:nw {#1}[]
2915     }
2916   }
2917 }
2918 }
2919
2920
2921 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2922   \exp_args:Nnx \use:nn {
2923     \str_set:Nn \l_stex_current_symbol_str { #1 }
2924     \bool_set_false:N \l_stex_allow_semantic_bool
2925     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2926       \comp{ #2 }
2927     }
2928   }{
2929     \__stex_terms_reset:N \l_stex_current_symbol_str
2930     \bool_set_true:N \l_stex_allow_semantic_bool
2931   }
2932 }
2933
2934 \cs_new_protected:Nn \__stex_terms_find_notation:nn {
2935   \__stex_terms_args:n { #2 }
2936   \seq_if_empty:cTF {
2937     l_stex_symdecl_ #1 _notations
2938   } {
2939     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2940   } {
2941     \bool_lazy_all:nTF {
2942       {\str_if_empty_p:N \l__stex_terms_variant_str}
2943       {\str_if_empty_p:N \l__stex_terms_lang_str}
2944     }{
2945       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l__stex_terms_variant_str
2946     }{
2947       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
2948         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2949       }{
2950         \str_set:Nx \l__stex_terms_variant_str { \l__stex_terms_variant_str \c_hash_str \l__
2951       }{
2952         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2953           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```



```

2954     }
2955   }
2956 }
2957 }
2958 }
2959
2960 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
2961   \exp_args:Nnx \use:nn {
2962     \str_set:Nn \l_stex_current_symbol_str { #1 }
2963     \__stex_terms_find_notation:nn { #1 }{ #2 }
2964     \bool_set_false:N \l_stex_allow_semantic_bool
2965     \cs_if_exist:cTF {
2966       stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2967     }{
2968       \use:c{stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2969     }{
2970       \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str}
2971     }
2972   }{
2973     \__stex_terms_reset:N \l_stex_current_symbol_str
2974     \bool_set_true:N \l_stex_allow_semantic_bool
2975   }
2976 }
2977
2978 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
2979   \__stex_terms_find_notation:nn { #1 }{ #2 }
2980   \cs_if_exist:cTF {
2981     stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2982   }{
2983     \tl_set:Nx \stex_symbol_after_invokation_tl {
2984       \__stex_terms_reset:N \stex_symbol_after_invokation_tl
2985       \__stex_terms_reset:N \l_stex_current_symbol_str
2986       \bool_set_true:N \l_stex_allow_semantic_bool
2987     }
2988     \str_set:Nn \l_stex_current_symbol_str { #1 }
2989     \bool_set_false:N \l_stex_allow_semantic_bool
2990     \use:c{stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2991   }{
2992     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2993       ~\l__stex_terms_variant_str
2994     }
2995   }
2996 }
2997
2998 \prop_new:N \l__stex_terms_custom_args_prop
2999
3000 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3001   \exp_args:Nnx \use:nn {
3002     \bool_set_false:N \l_stex_allow_semantic_bool
3003     \str_set:Nn \l_stex_current_symbol_str { #1 }
3004     \prop_clear:N \l__stex_terms_custom_args_prop
3005     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3006     \prop_get:cnN {
3007       l_stex_symdecl_#1 _prop

```

```

3008     }{ args } \l_tmpa_str
3009     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3010     \tl_set:Nn \arg { \__stex_terms_arg: }
3011     \str_if_empty:NTF \l_tmpa_str {
3012         \_stex_term_oms:nnn {#1}{#1}{#2}
3013     }{
3014         \str_if_in:NnTF \l_tmpa_str b {
3015             \_stex_term_ombind:nnn {#1}{#1}{#2}
3016         }{
3017             \str_if_in:NnTF \l_tmpa_str B {
3018                 \_stex_term_ombind:nnn {#1}{#1}{#2}
3019             }{
3020                 \_stex_term_oma:nnn {#1}{#1}{#2}
3021             }
3022         }
3023     }
3024     % TODO check that all arguments exist
3025 }{
3026     \__stex_terms_reset:N \l_stex_current_symbol_str
3027     \__stex_terms_reset:N \arg
3028     \__stex_terms_reset:N \l__stex_terms_custom_args_prop
3029     \bool_set_true:N \l_stex_allow_semantic_bool
3030 }
3031 }
3032
3033 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3034     \tl_if_empty:nTF {#2}{
3035         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3036         \bool_set_true:N \l_tmpa_bool
3037         \bool_do_while:Nn \l_tmpa_bool {
3038             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3039                 \int_incr:N \l_tmpa_int
3040             }{
3041                 \bool_set_false:N \l_tmpa_bool
3042             }
3043         }
3044     }{
3045         \int_set:Nn \l_tmpa_int { #2 }
3046         \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3047             % TODO throw error
3048         }
3049     }
3050     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3051     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3052         % TODO throw error
3053     }
3054     \bool_set_true:N \l_stex_allow_semantic_bool
3055     \IfBooleanTF#1{
3056         \stex_annotate_invisible:n {
3057             \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3058         }
3059     }{
3060         \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3061     }

```

```

3062 \bool_set_false:N \l_stex_allow_semantic_bool
3063 }
3064
3065
3066 \cs_new_protected:Nn \_stex_term_arg:nn {
3067 \bool_set_true:N \l_stex_allow_semantic_bool
3068 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3069 \bool_set_false:N \l_stex_allow_semantic_bool
3070 }
3071
3072 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3073 \exp_args:Nnx \use:nn
3074 { \int_set:Nn \l__stex_terms_downprec { #2 }
3075 \_stex_term_arg:nn { #1 }{ #3 }
3076 }
3077 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3078 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3079 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3080 % TODO sequences
3081 \clist_set:Nn \l_tmpa_clist{ #3 }
3082 \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3083 \tl_set:Nn \l_tmpa_tl { #3 }
3084 }{
3085 \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3086 \clist_reverse:N \l_tmpa_clist
3087 \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3088
3089 \clist_map_inline:Nn \l_tmpa_clist {
3090 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3091 \exp_args:Nno
3092 \l_tmpa_cs { ##1 } \l_tmpa_tl
3093 }
3094 }
3095 }
3096 \exp_args:Nnno
3097 \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3098 }

```

(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
3099 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3100 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3101 \int_new:N \l__stex_terms_downprec
3102 \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`

```
3103 \tl_set:Nn \l__stex_terms_left_bracket_str (
3104 \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

```
3105 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3106   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3107     \bool_set_false:N \l__stex_terms_brackets_done_bool
3108     #2
3109   } {
3110     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3111       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3112         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3113         \dobrackets { #2 }
3114       }
3115     }{ #2 }
3116   }
3117 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
3118 \bool_new:N \l__stex_terms_brackets_done_bool
3119 %\RequirePackage{scalerel}
3120 \cs_new_protected:Npn \dobrackets #1 {
3121   %\ThisStyle{\if D\m@switch
3122   %   \exp_args:Nnx \use:nn
3123   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3124   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3125   %   \else
3126   %   \exp_args:Nnx \use:nn
3127   %   {
3128     \bool_set_true:N \l__stex_terms_brackets_done_bool
3129     \int_set:Nn \l__stex_terms_downprec \infprec
3130     \l__stex_terms_left_bracket_str
3131     #1
3132   }
3133   {
3134     \bool_set_false:N \l__stex_terms_brackets_done_bool
3135     \l__stex_terms_right_bracket_str
3136     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3137   }
3138   %\fi}
3139 }
```

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

\withbrackets

```
3140 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3141   \exp_args:Nnx \use:nn
3142   {
3143     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3144     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3145     #3
3146   }
3147   {
3148     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3149     {\l__stex_terms_left_bracket_str}
3150     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3151     {\l__stex_terms_right_bracket_str}
3152   }
3153 }
```

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

\STEXinvisible

```
3154 \cs_new_protected:Npn \STEXinvisible #1 {
3155   \stex_annotate_invisible:n { #1 }
3156 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3157 \cs_new_protected:Nn \_stex_term_oms:nnn {
3158   \stex_annotate:nnn{ OMID }{ #2 }{
3159     \stex_highlight_term:nn { #1 } { #3 }
3160   }
3161 }
3162
3163 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3164   \__stex_terms_maybe_brackets:nn { #3 }{
3165     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3166   }
3167 }
```

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

_stex_term_math_omv:nn

```
3168 \cs_new_protected:Nn \_stex_term_omv:nn {
3169   \stex_annotate:nnn{ OMID }{ #1 }{
3170     \stex_highlight_term:nn { #1 } { #2 }
3171   }
3172 }
```

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

_stex_term_math_oma:nnnn

```
3173 \cs_new_protected:Nn \_stex_term_oma:nnn {
3174   \stex_annotate:nnn{ OMA }{ #2 }{
3175     \stex_highlight_term:nn { #1 } { #3 }
3176   }
```

```

3177 }
3178
3179 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3180   \__stex_terms_maybe_brackets:nn { #3 }{
3181     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3182   }
3183 }

```

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

`\stex_term_math_omb:nnnn`

```

3184 \cs_new_protected:Nn \stex_term_ombind:nnn {
3185   \stex_annotate:nnn{ OMBIND }{ #2 }{
3186     \stex_highlight_term:nn { #1 } { #3 }
3187   }
3188 }
3189
3190 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3191   \__stex_terms_maybe_brackets:nn { #3 }{
3192     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3193   }
3194 }

```

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

`\stex_term_custom:nn`

```

3195 \cs_new_protected:Nn \stex_term_custom:nn {
3196   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3197   \str_set:Nn \l_tmpa_str { #2 }
3198   \tl_clear:N \l_tmpa_tl
3199   \int_zero:N \l_tmpa_int
3200   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3201   \__stex_terms_custom_loop:
3202 }

```

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

`__stex_terms_custom_loop:`

```

3203 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3204   \bool_set_false:N \l_tmpa_bool
3205   \bool_while_do:nn {
3206     \str_if_eq_p:ee X {
3207       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3208     }
3209   }{
3210     \int_incr:N \l_tmpa_int
3211   }
3212
3213   \peek_charcode:NTF [ {
3214     % notation/text component
3215     \__stex_terms_custom_component:w
3216   } {
3217     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3218       % all arguments read => finish
3219       \__stex_terms_custom_final:

```

```

3220 } {
3221   % arguments missing
3222   \peek_charcode_remove:NTF * {
3223     % invisible, specific argument position or both
3224     \peek_charcode:NTF [ {
3225       % visible specific argument position
3226       \__stex_terms_custom_arg:wn
3227     } {
3228       % invisible
3229       \peek_charcode_remove:NTF * {
3230         % invisible specific argument position
3231         \__stex_terms_custom_arg_inv:wn
3232       } {
3233         % invisible next argument
3234         \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3235       }
3236     }
3237   } {
3238     % next normal argument
3239     \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3240   }
3241 }
3242 }
3243 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

3244 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3245   \bool_set_true:N \l_tmpa_bool
3246   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
3247 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

3248 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
3249   \str_set:Nx \l_tmpb_str {
3250     \str_item:Nn \l_tmpa_str { #1 }
3251   }
3252   \str_case:VnTF \l_tmpb_str {
3253     { X } {
3254       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3255     }
3256     { i } { \__stex_terms_custom_set_X:n { #1 } }
3257     { b } { \__stex_terms_custom_set_X:n { #1 } }
3258     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3259     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3260   }{}{
3261     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3262   }
3263
3264   \bool_if:nTF \l_tmpa_bool {
3265     \tl_put_right:Nx \l_tmpa_tl {
3266       \stex_annotate_invisible:n {

```

```

3267         \stex_term_arg:nn { \int_eval:n { #1 } }
3268         \exp_not:n { { #2 } }
3269     }
3270 }
3271 } {
3272     \tl_put_right:Nx \l_tmpa_tl {
3273         \stex_term_arg:nn { \int_eval:n { #1 } }
3274         \exp_not:n { { #2 } }
3275     }
3276 }
3277
3278 \__stex_terms_custom_loop:
3279 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

3280 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
3281     \str_set:Nx \l_tmpa_str {
3282         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3283         X
3284         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3285     }
3286 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

3287 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
3288     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3289     \__stex_terms_custom_loop:
3290 }

```

(End definition for __stex_terms_custom_component:.)

__stex_terms_custom_final:

```

3291 \cs_new_protected:Nn \__stex_terms_custom_final: {
3292     \int_compare:nNnTF \l_tmpb_int = 0 {
3293         \exp_args:Nnno \stex_term_oms:nnn
3294     }{
3295         \str_if_in:NnTF \l_tmpa_str {b} {
3296             \exp_args:Nnno \stex_term_ombind:nnn
3297         } {
3298             \exp_args:Nnno \stex_term_oma:nnn
3299         }
3300     }
3301     { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3302 }

```

(End definition for __stex_terms_custom_final:.)

\symref

\symname

```

3303 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3304
3305 \keys_define:nn { stex / symname } {

```



```

3306 pre .tl_set_x:N = \l__stex_terms_pre_tl ,
3307 post .tl_set_x:N = \l__stex_terms_post_tl ,
3308 root .tl_set_x:N = \l__stex_terms_root_tl
3309 }
3310
3311 \cs_new_protected:Nn \stex_symname_args:n {
3312 \tl_clear:N \l__stex_terms_post_tl
3313 \tl_clear:N \l__stex_terms_pre_tl
3314 \tl_clear:N \l__stex_terms_root_str
3315 \keys_set:nn { stex / symname } { #1 }
3316 }
3317
3318 \NewDocumentCommand \symref { m m }{
3319 \let\compemph_uri_prev:\compemph@uri
3320 \let\compemph@uri\symrefemph@uri
3321 \STEXsymbol{#1}!{ #2 }
3322 \let\compemph@uri\compemph_uri_prev:
3323 }
3324
3325 \NewDocumentCommand \synonym { 0{} m m }{
3326 \stex_symname_args:n { #1 }
3327 \let\compemph_uri_prev:\compemph@uri
3328 \let\compemph@uri\symrefemph@uri
3329 % TODO
3330 \STEXsymbol{#2}!\{ \l__stex_terms_pre_tl #3 \l__stex_terms_post_tl }
3331 \let\compemph@uri\compemph_uri_prev:
3332 }
3333
3334 \NewDocumentCommand \symname { 0{} m }{
3335 \stex_symname_args:n { #1 }
3336 \stex_get_symbol:n { #2 }
3337 \str_set:Nx \l_tmpa_str {
3338 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3339 }
3340 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3341
3342 \let\compemph_uri_prev:\compemph@uri
3343 \let\compemph@uri\symrefemph@uri
3344 \exp_args:NNx \use:nn
3345 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3346 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3347 } }
3348 \let\compemph@uri\compemph_uri_prev:
3349 }
3350
3351 \NewDocumentCommand \Symname { 0{} m }{
3352 \stex_symname_args:n { #1 }
3353 \stex_get_symbol:n { #2 }
3354 \str_set:Nx \l_tmpa_str {
3355 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3356 }
3357 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3358 \let\compemph_uri_prev:\compemph@uri
3359 \let\compemph@uri\symrefemph@uri

```

```

3360 \exp_args:NNx \use:nn
3361 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3362   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3363   \l__stex_terms_post_tl
3364 } }
3365 \let\compemph@uri\compemph_uri_prev:
3366 }

```

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

31.3 Notation Components

```

3367 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3368 \cs_new_protected:Nn \stex_highlight_term:nn {
3369   #2
3370 }
3371
3372 \cs_new_protected:Nn \stex_unhighlight_term:n {
3373   % \latexml_if:TF {
3374   %   #1
3375   % } {
3376   %   \rustex_if:TF {
3377   %     #1
3378   %   } {
3379     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3380   %   }
3381   % }
3382 }

```

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

```

\comp
\compemph@uri 3383 \cs_new_protected:Npn \comp #1 {
\compemph 3384   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3385     \rustex_if:TF {
\defemph@uri 3386       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3387     }{
\symrefemph@uri 3388       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3389     }
3390   }
3391 }
3392
3393 \cs_new_protected:Npn \compemph@uri #1 #2 {
3394   \compemph{ #1 }
3395 }
3396
3397
3398 \cs_new_protected:Npn \compemph #1 {
3399   #1
3400 }
3401
3402 \cs_new_protected:Npn \defemph@uri #1 #2 {

```

```

3403     \defemph{#1}
3404 }
3405
3406 \cs_new_protected:Npn \defemph #1 {
3407     \textbf{#1}
3408 }
3409
3410 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3411     \symrefemph{#1}
3412 }
3413
3414 \cs_new_protected:Npn \symrefemph #1 {
3415     \textbf{#1}
3416 }

```

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

\ellipses

```

3417 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3418 \bool_new:N \l_stex_inarray_bool
3419 \bool_set_false:N \l_stex_inarray_bool
3420 \NewDocumentCommand \parray { m m } {
3421     \begingroup
3422     \bool_set_true:N \l_stex_inarray_bool
3423     \begin{array}{#1}
3424         #2
3425     \end{array}
3426 \endgroup
3427 }
3428
3429 \NewDocumentCommand \prmatrix { m } {
3430     \begingroup
3431     \bool_set_true:N \l_stex_inarray_bool
3432     \begin{matrix}
3433         #1
3434     \end{matrix}
3435 \endgroup
3436 }
3437
3438 \def \maybepline {
3439     \bool_if:NT \l_stex_inarray_bool {\hline}
3440 }
3441
3442 \def \parrayline #1 #2 {
3443     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3444 }
3445
3446 \def \pmrow #1 { \parrayline{}{ #1 } }
3447
3448 \def \parraylineh #1 #2 {
3449     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}

```

```

3450 }
3451
3452 \def \parraycell #1 {
3453   #1 \bool_if:NT \l_stex_inarray_bool {&}
3454 }

```

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

31.4 Variables

```

3455 \@@=stex_variables\

```

\stex_invoke_variable:n Invokes a variable

```

3456 \cs_new_protected:Nn \stex_invoke_variable:n {
3457   \if_mode_math:
3458     \exp_after:wN \__stex_variables_invoke_math:n
3459   \else:
3460     \exp_after:wN \__stex_variables_invoke_text:n
3461   \fi: {#1}
3462 }
3463
3464 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3465   %TODO
3466 }
3467
3468
3469 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3470   \peek_charcode_remove:NTF ! {
3471     \peek_charcode_remove:NTF ! {
3472       \peek_charcode:NTF [ {
3473         \__stex_variables_invoke_op_custom:nw
3474       }{
3475         % TODO throw error
3476       }
3477     }{
3478       \__stex_variables_invoke_op:n { #1 }
3479     }
3480   }{
3481     \peek_charcode_remove:NTF * {
3482       \__stex_variables_invoke_text:n { #1 }
3483     }{
3484       \__stex_variables_invoke_math_ii:n { #1 }
3485     }
3486   }
3487 }
3488
3489 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3490   \cs_if_exist:cTF {
3491     stex_var_op_notation_ #1 _cs
3492   }{
3493     \exp_args:Nnx \use:nn {
3494       \str_set:Nn \l_stex_current_symbol_str { #1 }
3495       \use:c{stex_var_op_notation_ #1 _cs }
3496     }{

```

```

3497     \str_set:Nn \exp_not:N \l_stex_current_symbol_str {\l_stex_current_symbol_str}
3498   }
3499   }{
3500     \msg_error:nnxx{stex}{error/noop}{variable-#1}{}
3501   }
3502 }
3503
3504 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3505   \cs_if_exist:cTF {
3506     stex_var_notation_#1_cs
3507   }{
3508     \tl_set:Nx \stex_symbol_after_invokation_tl {
3509       \__stex_variables_reset:N \stex_symbol_after_invokation_tl
3510       \__stex_variables_reset:N \l_stex_current_symbol_str
3511       \bool_set_true:N \l_stex_allow_semantic_bool
3512     }
3513     \str_set:Nn \l_stex_current_symbol_str { #1 }
3514     \bool_set_false:N \l_stex_allow_semantic_bool
3515     \use:c{stex_var_notation_#1_cs}
3516   }{
3517     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
3518   }
3519 }

```

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

```

3520 </package>

```

Chapter 32

STEX -Structural Features Implementation

```
3521 ⟨*package⟩
3522
3523 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3524
3525 ⟨@@=stex_features⟩
    Warnings and error messages
3526 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3527   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3528 }
3529 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3530   Symbol~#1~not~assigned~in~interpretmodule~#2
3531 }
3532
```

32.1 Imports with modification

```
3533 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3534   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3535     \__stex_features_get_symbol_from_cs:n { #1 }
3536   }{
3537     % argument is a string
3538     % is it a command name?
3539     \cs_if_exist:cTF { #1 }{
3540       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3541       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3542       \str_if_empty:NTF \l_tmpa_str {
3543         \exp_args:Nx \cs_if_eq:NNTF {
3544           \tl_head:N \l_tmpa_tl
3545         } \stex_invoke_symbol:n {
3546           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3547         }{
3548           \__stex_features_get_symbol_from_string:n { #1 }

```

```

3549     }
3550   } {
3551     \__stex_features_get_symbol_from_string:n { #1 }
3552   }
3553   }{
3554     % argument is not a command name
3555     \__stex_features_get_symbol_from_string:n { #1 }
3556     % \l_stex_all_symbols_seq
3557   }
3558 }
3559 }
3560
3561 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3562   \str_set:Nn \l_tmpa_str { #1 }
3563   \bool_set_false:N \l_tmpa_bool
3564   \bool_if:NF \l_tmpa_bool {
3565     \tl_set:Nn \l_tmpa_tl {
3566       \msg_set:nnn{stex}{error/unknownsymbol}{
3567         No~symbol~#1~found!
3568       }
3569       \msg_error:nn{stex}{error/unknownsymbol}
3570     }
3571     \str_set:Nn \l_tmpa_str { #1 }
3572     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3573     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3574       \str_set:Nn \l_tmpb_str { ##1 }
3575       \str_if_eq:eeT { \l_tmpa_str } {
3576         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3577       } {
3578         \seq_map_break:n {
3579           \tl_set:Nn \l_tmpa_tl {
3580             \str_set:Nn \l_stex_get_symbol_uri_str {
3581               ##1
3582             }
3583             \__stex_features_get_symbol_check:
3584           }
3585         }
3586       }
3587     }
3588     \l_tmpa_tl
3589   }
3590 }
3591
3592 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3593   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3594   { \tl_tail:N \l_tmpa_tl }
3595   \tl_if_single:NTF \l_tmpa_tl {
3596     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3597       \exp_after:wN \str_set:Nn \exp_after:wN
3598       \l_stex_get_symbol_uri_str \l_tmpa_tl
3599       \__stex_features_get_symbol_check:
3600     }{
3601       % TODO
3602       % tail is not a single group

```

```

3603     }
3604 }{
3605     % TODO
3606     % tail is not a single group
3607 }
3608 }
3609
3610 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3611     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3612     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3613         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3614         \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3615         \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3616             \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3617                 \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3618             }
3619         }
3620     }{
3621         \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3622             \l_stex_current_copymodule_name_str~(inexplicably)
3623         }
3624     }
3625 }
3626
3627 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3628     \stex_import_module_uri:nn { #1 } { #2 }
3629     \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3630     \stex_import_require_module:nnnn
3631     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3632     { \l_stex_import_path_str } { \l_stex_import_name_str }
3633     \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3634     \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3635     \seq_clear:N \l__stex_features_copymodule_fields_seq
3636     \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3637         \seq_map_inline:cn {c_stex_module_###1_constants}{
3638             \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3639                 ###1 ? #####1
3640             }
3641         }
3642     }
3643     \seq_clear:N \l_tmpa_seq
3644     \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3645         name      = \l_stex_current_copymodule_name_str ,
3646         module    = \l_stex_current_module_str ,
3647         from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3648         includes  = \l_tmpa_seq ,
3649         fields    = \l_tmpa_seq
3650     }
3651     \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3652         as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3653     \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3654     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3655     \stex_if_smsmode:F {
3656         \begin{stex_annotate_env} {#4} {

```



```

3657     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3658   }
3659   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3660 }
3661 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3662 \bool_set_false:N \stex_html_do_output_bool
3663 }
3664 \cs_new_protected:Nn \stex_copymodule_end:n {
3665   \def \l_tmpa_cs ##1 ##2 {#1}
3666   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3667   \tl_clear:N \l_tmpa_tl
3668   \tl_clear:N \l_tmpb_tl
3669   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3670   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3671     \seq_map_inline:cn {c_stex_module_##1_constants}{
3672       \tl_clear:N \l_tmpc_tl
3673       \l_tmpa_cs{##1}{####1}
3674       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3675         \tl_put_right:Nx \l_tmpa_tl {
3676           \prop_set_from_keyval:cn {
3677             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3678           }{
3679             \exp_after:wN \prop_to_keyval:N \csname
3680               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3681             \endcsname
3682           }
3683           \seq_clear:c {
3684             l_stex_symdecl_
3685             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3686             _notations
3687           }
3688         }
3689         \tl_put_right:Nx \l_tmpc_tl {
3690           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3691           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3692         }
3693         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3694         \str_if_exist:cT {\l__stex_features_copymodule_##1?####1_macroname_str} {
3695           \tl_put_right:Nx \l_tmpc_tl {
3696             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3697           }
3698           \tl_put_right:Nx \l_tmpa_tl {
3699             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3700             \stex_invoke_symbol:n {
3701               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3702             }
3703           }
3704         }
3705       }
3706     }{
3707       \tl_put_right:Nx \l_tmpc_tl {
3708         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3709       }
3710       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3711 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / #####1 }
3712 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3713 \tl_put_right:Nx \l_tmpa_tl {
3714   \prop_set_from_keyval:cn {
3715     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3716   }{
3717     \prop_to_keyval:N \l_tmpa_prop
3718   }
3719   \seq_clear:c {
3720     l_stex_symdecl_
3721     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3722     _notations
3723   }
3724 }
3725 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3726 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3727   \tl_put_right:Nx \l_tmpc_tl {
3728     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3729   }
3730   \tl_put_right:Nx \l_tmpa_tl {
3731     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3732       \stex_invoke_symbol:n {
3733         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
3734       }
3735     }
3736   }
3737 }
3738 }
3739 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3740   \tl_put_right:Nx \l_tmpc_tl {
3741     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3742   }
3743 }
3744 \tl_put_right:Nx \l_tmpb_tl {
3745   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3746 }
3747 }
3748 }
3749 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3750 \tl_put_left:Nx \l_tmpa_tl {
3751   \prop_set_from_keyval:cn {
3752     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3753   }{
3754     \prop_to_keyval:N \l_stex_current_copymodule_prop
3755   }
3756 }
3757 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3758 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3759 \exp_args:Nx \stex_do_up_to_module:n {
3760   \exp_args:No \exp_not:n \l_tmpa_tl
3761 }
3762 \l_tmpb_tl
3763 \stex_if_smsmode:F {
3764   \end{stex_annotate_env}

```

```

3765 }
3766 }
3767
3768 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3769   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3770   \stex_deactivate_macro:Nn \symdecl {module~environments}
3771   \stex_deactivate_macro:Nn \symdef {module~environments}
3772   \stex_deactivate_macro:Nn \notation {module~environments}
3773   \stex_reactivate_macro:N \assign
3774   \stex_reactivate_macro:N \renamedec1
3775   \stex_reactivate_macro:N \donotcopy
3776   \stex_smsmode_do:
3777 }{
3778   \stex_copymodule_end:n {}
3779 }
3780
3781 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3782   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3783   \stex_deactivate_macro:Nn \symdecl {module~environments}
3784   \stex_deactivate_macro:Nn \symdef {module~environments}
3785   \stex_deactivate_macro:Nn \notation {module~environments}
3786   \stex_reactivate_macro:N \assign
3787   \stex_reactivate_macro:N \renamedec1
3788   \stex_reactivate_macro:N \donotcopy
3789   \stex_smsmode_do:
3790 }{
3791   \stex_copymodule_end:n {
3792     \tl_if_exist:cF {
3793       l__stex_features_copymodule_##1?##2_def_tl
3794     }{
3795       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3796         ##1?##2
3797       }{\l_stex_current_copymodule_name_str}
3798     }
3799   }
3800 }
3801
3802 \NewDocumentCommand \donotcopy { 0{} m}{
3803   \stex_import_module_uri:nn { #1 } { #2 }
3804   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3805   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3806     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3807     \seq_map_inline:cn {c_stex_module_##1_constants}{
3808       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3809       \bool_lazy_any_p:nT {
3810         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3811         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3812         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3813       }{
3814         % TODO throw error
3815       }
3816     }
3817   }
3818 }

```

```

3819 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3820 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3821 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3822 }
3823
3824 \NewDocumentCommand \assign { m m }{
3825   \stex_get_symbol_in_copymodule:n {#1}
3826   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3827   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3828 }
3829
3830 \keys_define:nn { stex / renamedec1 } {
3831   name .str_set_x:N = \l_stex_renamedec1_name_str
3832 }
3833 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3834   \str_clear:N \l_stex_renamedec1_name_str
3835
3836   \keys_set:nn { stex / renamedec1 } { #1 }
3837 }
3838
3839 \NewDocumentCommand \renamedec1 { 0{} m m }{
3840   \__stex_features_renamedec1_args:n { #1 }
3841   \stex_get_symbol_in_copymodule:n {#2}
3842   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3843   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3844   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3845     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3846       \l_stex_get_symbol_uri_str
3847     } }
3848   } {
3849     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3850     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3851     \prop_set_eq:cc {l_stex_symdecl_
3852       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3853     _prop
3854     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3855     \seq_set_eq:cc {l_stex_symdecl_
3856       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3857     _notations
3858     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3859     \prop_put:cnx {l_stex_symdecl_
3860       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3861     _prop
3862     }{ name }{ \l_stex_renamedec1_name_str }
3863     \prop_put:cnx {l_stex_symdecl_
3864       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3865     _prop
3866     }{ module }{ \l_stex_current_module_str }
3867     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3868       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3869     }
3870     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3871       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3872     } }

```

```

3873 }
3874 }
3875
3876 \stex_deactivate_macro:Nn \assign {copymodules}
3877 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3878 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3879
3880
3881 \seq_new:N \l_stex_implicit_morphisms_seq
3882 \NewDocumentCommand \implicitmorphism { O{} m m }{
3883   \stex_import_module_uri:nn { #1 } { #2 }
3884   \stex_debug:nn{implicits}{
3885     Implicit~morphism:~
3886     \l_stex_module_ns_str ? \l__stex_features_name_str
3887   }
3888   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3889     \l_stex_module_ns_str ? \l__stex_features_name_str
3890   }{
3891     \msg_error:nnn{stex}{error/conflictingmodules}{
3892       \l_stex_module_ns_str ? \l__stex_features_name_str
3893     }
3894   }
3895
3896   % TODO
3897
3898
3899
3900   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3901     \l_stex_module_ns_str ? \l__stex_features_name_str
3902   }
3903 }
3904

```

32.2 The feature environment

structural@feature

```

3905
3906 \NewDocumentEnvironment{structural@feature}{ m m m }{
3907   \stex_if_in_module:F {
3908     \msg_set:nnn{stex}{error/nomodule}{
3909       Structural~Feature~has~to~occur~in~a~module:\\
3910       Feature~#2~of~type~#1\\
3911       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3912     }
3913     \msg_error:nn{stex}{error/nomodule}
3914   }
3915
3916   \str_set:Nx \l_stex_module_name_str {
3917     \prop_item:Nn \l_stex_current_module_prop
3918     { name } / #2 - feature
3919   }
3920
3921   \str_set:Nx \l_stex_module_ns_str {

```

```

3922     \prop_item:Nn \l_stex_current_module_prop
3923     { ns }
3924 }
3925
3926
3927 \str_clear:N \l_tmpa_str
3928 \seq_clear:N \l_tmpa_seq
3929 \tl_clear:N \l_tmpa_tl
3930 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3931   origname = #2,
3932   name      = \l_stex_module_name_str ,
3933   ns        = \l_stex_module_ns_str ,
3934   imports   = \exp_not:o { \l_tmpa_seq } ,
3935   constants = \exp_not:o { \l_tmpa_seq } ,
3936   content   = \exp_not:o { \l_tmpa_tl } ,
3937   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3938   lang      = \l_stex_module_lang_str ,
3939   sig       = \l_tmpa_str ,
3940   meta      = \l_tmpa_str ,
3941   feature   = #1 ,
3942 }
3943
3944 \stex_if_smsmode:F {
3945   \begin{stex_annotate_env}{ feature:#1 }{}
3946   \stex_annotate_invisible:nnn{header}{}{ #3 }
3947 }
3948 }{
3949   \str_set:Nx \l_tmpa_str {
3950     c_stex_feature_
3951     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3952     \prop_item:Nn \l_stex_current_module_prop { name }
3953     _prop
3954   }
3955   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3956   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3957   \stex_if_smsmode:F {
3958     \end{stex_annotate_env}
3959   }
3960 }
3961

```

32.3 Features

structure

```

3962
3963 \prop_new:N \l_stex_all_structures_prop
3964
3965 \keys_define:nn { stex / features / structure } {
3966   name .str_set_x:N = \l__stex_features_structure_name_str ,
3967 }
3968
3969 \cs_new_protected:Nn \__stex_features_structure_args:n {
3970   \str_clear:N \l__stex_features_structure_name_str

```

```

3971 \keys_set:nn { stex / features / structure } { #1 }
3972 }
3973
3974 \NewDocumentEnvironment{mathstructure}{ O{} m }{
3975   \__stex_features_structure_args:n { #1 }
3976   \str_if_empty:NT \l__stex_features_structure_name_str {
3977     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3978   }
3979   \exp_args:Nnnx
3980   \begin{structural@feature}{ structure }
3981     { \l__stex_features_structure_name_str }{}
3982     \seq_clear:N \l_tmpa_seq
3983     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3984     \stex_smsmode_do:
3985   }{
3986     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3987     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3988     \str_set:Nx \l_tmpa_str {
3989       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3990       \prop_item:Nn \l_stex_current_module_prop { name }
3991     }
3992     \seq_map_inline:Nn \l_tmpa_seq {
3993       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3994     }
3995     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3996     \exp_args:Nnx
3997     \AddToHookNext { env / mathstructure / after }{
3998       \symdecl{ #2 }[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3999         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{}{}
4000       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]
4001       \STEXexport {
4002         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4003           {\prop_item:Nn \l_stex_current_module_prop { origname }}
4004           {\l_tmpa_str}
4005         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4006           {#2}{\l_tmpa_str}
4007       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4008       %   \prop_item:Nn \l_stex_current_module_prop { origname },
4009       %   \l_tmpa_str
4010       % }
4011       % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4012       %   #2,\l_tmpa_str
4013       % }
4014       % \tl_set:cx { #2 } {
4015       %   \stex_invoke_structure:n { \l_tmpa_str }
4016       % }
4017     }
4018
4019   \end{structural@feature}
4020   % \g_stex_last_feature_prop
4021 }

```

\instantiate

```

4022 \seq_new:N \l__stex_features_structure_field_seq

```

```

4023 \str_new:N \l__stex_features_structure_field_str
4024 \str_new:N \l__stex_features_structure_def_tl
4025 \prop_new:N \l__stex_features_structure_prop
4026 \NewDocumentCommand \instantiate { m O{} m }{
4027   \prop_get:NnN \l__stex_all_structures_prop {#1} \l_tmpa_str
4028   \prop_set_eq:Nc \l__stex_features_structure_prop {
4029     c_stex_feature_\l_tmpa_str _prop
4030   }
4031   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
4032   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
4033     \seq_set_split:Nnn \l_tmpa_seq={}{ ##1 }
4034     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4035       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
4036       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
4037         {!} \l_tmpa_tl
4038       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4039         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
4040         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4041         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4042       }{
4043         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
4044         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4045         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
4046           \l_tmpa_tl
4047         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4048           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
4049           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4050         }{
4051           \tl_clear:N \l_tmpb_tl
4052         }
4053       }
4054     }{
4055       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
4056       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4057         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
4058         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
4059         \tl_clear:N \l_tmpa_tl
4060       }{
4061         % TODO throw error
4062       }
4063     }
4064     % \l_tmpa_str: name
4065     % \l_tmpa_tl: definiens
4066     % \l_tmpb_tl: notation
4067     \tl_if_empty:NT \l__stex_features_structure_field_str {
4068       % TODO throw error
4069     }
4070     \str_clear:N \l_tmpb_str
4071
4072     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4073     \seq_map_inline:Nn \l_tmpa_seq {
4074       \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
4075       \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
4076       \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {

```



```

4077         \seq_map_break:n {
4078             \str_set:Nn \l_tmpb_str { ####1 }
4079         }
4080     }
4081 }
4082 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
4083 \l_tmpb_str
4084
4085 \tl_if_empty:NTF \l_tmpb_tl {
4086     \tl_if_empty:NF \l_tmpa_tl {
4087         \exp_args:Nx \use:n {
4088             \symdecl{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4089         }
4090     }
4091 }{
4092     \tl_if_empty:NTF \l_tmpa_tl {
4093         \exp_args:Nx \use:n {
4094             \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str]\exp_after:wN\
4095         }
4096     }
4097 }{
4098     \exp_args:Nx \use:n {
4099         \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4100         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
4101     }
4102 }
4103 }
4104 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
4105 % \prop_item:Nn \l_stex_current_module_prop {name} ?
4106 % #3/\l__stex_features_structure_field_str
4107 % \par
4108 % \expandafter\present\csname
4109 %     l_stex_symdecl_
4110 %     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4111 %     \prop_item:Nn \l_stex_current_module_prop {name} ?
4112 %     #3/\l__stex_features_structure_field_str
4113 %     _prop
4114 % \endcsname
4115 }
4116
4117 \tl_clear:N \l__stex_features_structure_def_tl
4118
4119 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4120 \seq_map_inline:Nn \l_tmpa_seq {
4121     \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4122     \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4123     \exp_args:Nx \use:n {
4124         \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
4125     }
4126 }
4127 }
4128
4129 \prop_if_exist:cF {
4130     l_stex_symdecl_

```

```

4131     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4132     \prop_item:Nn \l_stex_current_module_prop {name} ?
4133     #3/\l_tmpa_str
4134     _prop
4135   }{
4136     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
4137     \l_tmpb_str
4138     \exp_args:Nx \use:n {
4139       \symdecl{#3/\l_tmpa_str}[args=\l_tmpb_str]
4140     }
4141   }
4142 }
4143
4144 \symdecl*{#3}[type={\STEXsymbol{module-type}}{
4145   \_stex_term_math_oms:nnnn {
4146     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4147     \prop_item:Nn \l__stex_features_structure_prop {name}
4148     }{}{0}{}
4149   }]]
4150
4151 % TODO: -> sms file
4152
4153 \tl_set:cx{ #3 }{
4154   \stex_invoke_structure:nnn {
4155     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4156     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
4157   } {
4158     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4159     \prop_item:Nn \l__stex_features_structure_prop {name}
4160   }
4161 }
4162 \stex_smsmode_do:
4163 }

```

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

\stex_invoke_structure:nnn

```

4164 % #1: URI of the instance
4165 % #2: URI of the instantiated module
4166 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4167   \tl_if_empty:nTF{ #3 }{
4168     \prop_set_eq:Nc \l__stex_features_structure_prop {
4169       c_stex_feature_ #2 _prop
4170     }
4171     \tl_clear:N \l_tmpa_tl
4172     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4173     \seq_map_inline:Nn \l_tmpa_seq {
4174       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4175       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4176       \cs_if_exist:cT {
4177         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4178       }{
4179         \tl_if_empty:NF \l_tmpa_tl {
4180           \tl_put_right:Nn \l_tmpa_tl {,}

```

```

4181     }
4182     \tl_put_right:Nx \l_tmpa_tl {
4183       \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4184     }
4185   }
4186 }
4187 \exp_args:No \mathstruct \l_tmpa_tl
4188 }{
4189   \stex_invoke_symbol:n{#1/#3}
4190 }
4191 }

```

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

```

4192 </package>

```

Chapter 33

STEX -Statements Implementation

```
4193 <*package>
4194
4195 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4196
4197 <@@=stex_statements>
4198
4199 Warnings and error messages
4198
\titleemph
4199 \def\titleemph#1{\textbf{#1}}
4198
(End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

definiendum

```
4200 \keys_define:nn {stex / definiendum }{
4201   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4202   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4203   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4204   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4205 }
4206 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4207   \str_clear:N \l__stex_statements_definiendum_root_str
4208   \tl_clear:N \l__stex_statements_definiendum_post_tl
4209   \str_clear:N \l__stex_statements_definiendum_gfa_str
4210   \keys_set:nn { stex / definiendum }{ #1 }
4211 }
4212 \NewDocumentCommand \definiendum { O{} m m } {
4213   \__stex_statements_definiendum_args:n { #1 }
4214   \stex_get_symbol:n { #2 }
4215   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
4216   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4217     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4218     \tl_set:Nn \l_tmpa_tl { #3 }
4219   } {
4220     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4221     \tl_set:Nn \l_tmpa_tl {
4222       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4223     }
4224   }
4225 } {
4226   \tl_set:Nn \l_tmpa_tl { #3 }
4227 }
4228
4229 % TODO root
4230 \rustex_if:TF {
4231   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4232 } {
4233   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4234 }
4235 }
4236 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4237
4238 \NewDocumentCommand \definame { 0{ } m } {
4239   \__stex_statements_definiendum_args:n { #1 }
4240   % TODO: root
4241   \stex_get_symbol:n { #2 }
4242   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4243   \str_set:Nx \l_tmpa_str {
4244     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4245   }
4246   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4247   \rustex_if:TF {
4248     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4249       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4250     }
4251   } {
4252     \defemph@uri {
4253       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4254     } { \l_stex_get_symbol_uri_str }
4255   }
4256 }
4257 \stex_deactivate_macro:Nn \definame {definition~environments}
4258
4259 \NewDocumentCommand \Definame { 0{ } m } {
4260   \__stex_statements_definiendum_args:n { #1 }
4261   \stex_get_symbol:n { #2 }
4262   \str_set:Nx \l_tmpa_str {
4263     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4264   }
4265   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4266   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4267   \rustex_if:TF {

```

```

4268 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4269 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4270 }
4271 } {
4272 \defemph@uri {
4273 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4274 } { \l_stex_get_symbol_uri_str }
4275 }
4276 }
4277 \stex_deactivate_macro:Nn \Definame {definition~environments}
4278
4279 \NewDocumentCommand \premise { m }{
4280 \stex_annotate:nnn{ premise }{}{ #1 }
4281 }
4282 \NewDocumentCommand \conclusion { m }{
4283 \stex_annotate:nnn{ conclusion }{}{ #1 }
4284 }
4285 \NewDocumentCommand \definiens { m }{
4286 \stex_annotate:nnn{ definiens }{}{ #1 }
4287 }
4288
4289 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4290 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4291 \stex_deactivate_macro:Nn \definiens {definition~environments}
4292

```

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

sdefinition

```

4293
4294 \keys_define:nn {stex / sdefinition }{
4295 type .str_set_x:N = \sdefinitiontype,
4296 id .str_set_x:N = \sdefinitionid,
4297 name .str_set_x:N = \sdefinitionname,
4298 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4299 title .tl_set:N = \sdefinitiontitle
4300 }
4301 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4302 \str_clear:N \sdefinitiontype
4303 \str_clear:N \sdefinitionid
4304 \str_clear:N \sdefinitionname
4305 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4306 \tl_clear:N \sdefinitiontitle
4307 \keys_set:nn { stex / sdefinition }{}{ #1 }
4308 }
4309
4310 \NewDocumentEnvironment{sdefinition}{0{}}{
4311 \__stex_statements_sdefinition_args:n{ #1 }
4312 \stex_reactivate_macro:N \definiendum
4313 \stex_reactivate_macro:N \definame
4314 \stex_reactivate_macro:N \Definame
4315 \stex_reactivate_macro:N \premise
4316 \stex_reactivate_macro:N \definiens
4317 \stex_if_smsmode:F{

```

```

4318 \seq_clear:N \l_tmpa_seq
4319 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4320   \tl_if_empty:nF{ ##1 }{
4321     \stex_get_symbol:n { ##1 }
4322     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4323       \l_stex_get_symbol_uri_str
4324     }
4325   }
4326 }
4327 \exp_args:Nnnx
4328 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4329 \str_if_empty:NF \sdefinitiontype {
4330   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4331 }
4332 \clist_set:No \l_tmpa_clist \sdefinitiontype
4333 \tl_clear:N \l_tmpa_tl
4334 \clist_map_inline:Nn \l_tmpa_clist {
4335   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4336     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4337   }
4338 }
4339 \tl_if_empty:NTF \l_tmpa_tl {
4340   \__stex_statements_sdefinition_start:
4341 }{
4342   \l_tmpa_tl
4343 }
4344 }
4345 \stex_ref_new_doc_target:n \sdefinitionid
4346 \stex_smsmode_do:
4347 }{
4348   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4349   \stex_if_smsmode:F {
4350     \clist_set:No \l_tmpa_clist \sdefinitiontype
4351     \tl_clear:N \l_tmpa_tl
4352     \clist_map_inline:Nn \l_tmpa_clist {
4353       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4354         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4355       }
4356     }
4357     \tl_if_empty:NTF \l_tmpa_tl {
4358       \__stex_statements_sdefinition_end:
4359     }{
4360       \l_tmpa_tl
4361     }
4362     \end{stex_annotate_env}
4363   }
4364 }

```

\stexpatchdefinition

```

4365 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4366   \par\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
4367     ~(\sdefinitiontitle)
4368   }~}
4369 }

```

```

4370 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4371
4372 \newcommand\stexpatchdefinition[3] [] {
4373   \str_set:Nx \l_tmpa_str{ #1 }
4374   \str_if_empty:NTF \l_tmpa_str {
4375     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4376     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4377   }{
4378     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4379     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4380   }
4381 }

```

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

\inlinedef inline:

```

4382 \keys_define:nn {stex / inlinedef }{
4383   type      .str_set_x:N = \sdefinitiontype,
4384   id        .str_set_x:N = \sdefinitionid,
4385   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4386   name      .str_set_x:N = \sdefinitionname
4387 }
4388 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4389   \str_clear:N \sdefinitiontype
4390   \str_clear:N \sdefinitionid
4391   \str_clear:N \sdefinitionname
4392   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4393   \keys_set:nn { stex / inlinedef }{ #1 }
4394 }
4395 \NewDocumentCommand \inlinedef { 0{} m } {
4396   \beginngroup
4397   \__stex_statements_inlinedef_args:n{ #1 }
4398   \stex_reactivate_macro:N \definiendum
4399   \stex_reactivate_macro:N \definame
4400   \stex_reactivate_macro:N \Definame
4401   \stex_reactivate_macro:N \premise
4402   \stex_reactivate_macro:N \definiens
4403   \stex_ref_new_doc_target:n \sdefinitionid
4404   \stex_if_smsmode:TF{
4405     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4406   }{
4407     \seq_clear:N \l_tmpa_seq
4408     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4409       \tl_if_empty:nF{ ##1 }{
4410         \stex_get_symbol:n { ##1 }
4411         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4412           \l_stex_get_symbol_uri_str
4413         }
4414       }
4415     }
4416     \exp_args:Nnx
4417     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4418       \str_if_empty:NF \sdefinitiontype {
4419         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```



```

4420     }
4421     #2
4422     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4423   }
4424 }
4425 \endgroup
4426 \stex_smsmode_do:
4427 }

```

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

33.2 Assertions

sassertion

```

4428
4429 \keys_define:nn {stex / sassertion }{
4430   type      .str_set_x:N = \sassertiontype,
4431   id        .str_set_x:N = \sassertionid,
4432   title     .tl_set:N    = \sassertiontitle ,
4433   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4434   name      .str_set_x:N = \sassertionname
4435 }
4436 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4437   \str_clear:N \sassertiontype
4438   \str_clear:N \sassertionid
4439   \str_clear:N \sassertionname
4440   \clist_clear:N \l__stex_statements_sassertion_for_clist
4441   \tl_clear:N \sassertiontitle
4442   \keys_set:nn { stex / sassertion }{ #1 }
4443 }
4444
4445 %\tl_new:N \g__stex_statements_aftergroup_tl
4446
4447 \NewDocumentEnvironment{sassertion}{0{}}{
4448   \__stex_statements_sassertion_args:n{ #1 }
4449   \stex_reactivate_macro:N \premise
4450   \stex_reactivate_macro:N \conclusion
4451   \stex_if_smsmode:F {
4452     \seq_clear:N \l_tmpa_seq
4453     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4454       \tl_if_empty:nF{ ##1 }{
4455         \stex_get_symbol:n { ##1 }
4456         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4457           \l_stex_get_symbol_uri_str
4458         }
4459       }
4460     }
4461     \exp_args:Nnnx
4462     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4463     \str_if_empty:NF \sassertiontype {
4464       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4465     }
4466     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4467 \tl_clear:N \l_tmpa_tl
4468 \clist_map_inline:Nn \l_tmpa_clist {
4469   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4470     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4471   }
4472 }
4473 \tl_if_empty:NTF \l_tmpa_tl {
4474   \__stex_statements_sassertion_start:
4475 }{
4476   \l_tmpa_tl
4477 }
4478 }
4479 \str_if_empty:NTF \sassertionid {
4480   \str_if_empty:NF \sassertionname {
4481     \stex_ref_new_doc_target:n {}
4482   }
4483 } {
4484   \stex_ref_new_doc_target:n \sassertionid
4485 }
4486 \stex_smsmode_do:
4487 ){
4488   \str_if_empty:NF \sassertionname {
4489     \stex_symdecl_do:nn{ }\sassertionname}
4490     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4491   }
4492   \stex_if_smsmode:F {
4493     \clist_set:Nn \l_tmpa_clist \sassertiontype
4494     \tl_clear:N \l_tmpa_tl
4495     \clist_map_inline:Nn \l_tmpa_clist {
4496       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4497         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4498       }
4499     }
4500     \tl_if_empty:NTF \l_tmpa_tl {
4501       \__stex_statements_sassertion_end:
4502     }{
4503       \l_tmpa_tl
4504     }
4505     \end{stex_annotate_env}
4506   }
4507 }

```

\stexpatchassertion

```

4508
4509 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4510   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4511     (\sassertiontitle)
4512   }~}
4513 }
4514 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4515
4516 \newcommand\stexpatchassertion[3] [] {
4517   \str_set:Nx \l_tmpa_str{ #1 }
4518   \str_if_empty:NTF \l_tmpa_str {

```

```

4519     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4520     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4521   }{
4522     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4523     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4524   }
4525 }

```

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

\inlineass inline:

```

4526 \keys_define:nn {stex / inlineass }{
4527   type      .str_set_x:N = \sassertiontype,
4528   id        .str_set_x:N = \sassertionid,
4529   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4530   name      .str_set_x:N = \sassertionname
4531 }
4532 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4533   \str_clear:N \sassertiontype
4534   \str_clear:N \sassertionid
4535   \str_clear:N \sassertionname
4536   \clist_clear:N \l__stex_statements_sassertion_for_clist
4537   \keys_set:nn { stex / inlineass }{ #1 }
4538 }
4539 \NewDocumentCommand \inlineass { 0{} m } {
4540   \begin_group
4541     \stex_reactivate_macro:N \premise
4542     \stex_reactivate_macro:N \conclusion
4543     \__stex_statements_inlineass_args:n{ #1 }
4544     \str_if_empty:NTF \sassertionid {
4545       \str_if_empty:NF \sassertionname {
4546         \stex_ref_new_doc_target:n { }
4547       }
4548     } {
4549       \stex_ref_new_doc_target:n \sassertionid
4550     }
4551
4552     \stex_if_smsmode:TF{
4553       \str_if_empty:NF \sassertionname {
4554         \stex_symdecl_do:nn{}{\sassertionname}
4555         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4556       }
4557     }{
4558       \seq_clear:N \l_tmpa_seq
4559       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4560         \tl_if_empty:nF{ ##1 }{
4561           \stex_get_symbol:n { ##1 }
4562           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4563             \l_stex_get_symbol_uri_str
4564           }
4565         }
4566       }
4567       \exp_args:Nnx
4568       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4569     \str_if_empty:NF \sassertiontype {
4570       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4571   }
4572   #2
4573   \str_if_empty:NF \sassertionname {
4574     \stex_symdecl_do:nn{}{\sassertionname}
4575     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4576   }
4577 }
4578 }
4579 \endgroup
4580 \stex_smsmode_do:
4581 }

```

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

33.3 Examples

`sexample`

```

4582
4583 \keys_define:nn {stex / sexample }{
4584   type      .str_set_x:N = \exampletype,
4585   id        .str_set_x:N = \sexampleid,
4586   title     .tl_set:N    = \sexampletitle,
4587   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4588 }
4589 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4590   \str_clear:N \sexampletype
4591   \str_clear:N \sexampleid
4592   \tl_clear:N \sexampletitle
4593   \clist_clear:N \l__stex_statements_sexample_for_clist
4594   \keys_set:nn { stex / sexample }{ #1 }
4595 }
4596
4597 \NewDocumentEnvironment{sexample}{0{}}{
4598   \__stex_statements_sexample_args:n{ #1 }
4599   \stex_reactivate_macro:N \premise
4600   \stex_reactivate_macro:N \conclusion
4601   \stex_if_smsmode:F {
4602     \seq_clear:N \l_tmpa_seq
4603     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4604       \tl_if_empty:nF{ ##1 }{
4605         \stex_get_symbol:n { ##1 }
4606         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4607           \l_stex_get_symbol_uri_str
4608         }
4609       }
4610     }
4611     \exp_args:Nnnx
4612     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4613     \str_if_empty:NF \sexampletype {
4614       \stex_annotate_invisible:nnn{type}{\sexampletype}{}}
4615   }

```

```

4616 \clist_set:No \l_tmpa_clist \sexamplotype
4617 \tl_clear:N \l_tmpa_tl
4618 \clist_map_inline:Nn \l_tmpa_clist {
4619   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4620     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4621   }
4622 }
4623 \tl_if_empty:NTF \l_tmpa_tl {
4624   \__stex_statements_sexample_start:
4625 }{
4626   \l_tmpa_tl
4627 }
4628 }
4629 \str_if_empty:NF \sexampleid {
4630   \stex_ref_new_doc_target:n \sexampleid
4631 }
4632 \stex_smsmode_do:
4633 }{
4634   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4635   \stex_if_smsmode:F {
4636     \clist_set:No \l_tmpa_clist \sexamplotype
4637     \tl_clear:N \l_tmpa_tl
4638     \clist_map_inline:Nn \l_tmpa_clist {
4639       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4640         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4641       }
4642     }
4643     \tl_if_empty:NTF \l_tmpa_tl {
4644       \__stex_statements_sexample_end:
4645     }{
4646       \l_tmpa_tl
4647     }
4648     \end{stex_annotate_env}
4649   }
4650 }

```

\stexpatchexample

```

4651
4652 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4653   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4654     (\sexamplename)
4655   }~}
4656 }
4657 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4658
4659 \newcommand\stexpatchexample[3]{} {
4660   \str_set:Nx \l_tmpa_str{ #1 }
4661   \str_if_empty:NTF \l_tmpa_str {
4662     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4663     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4664   }{
4665     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4666     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4667   }

```

4668 }

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

`\inlineex` inline:

```

4669 \keys_define:nn {stex / inlineex }{
4670   type      .str_set_x:N = \sexamplotype,
4671   id        .str_set_x:N = \sexampleid,
4672   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4673   name      .str_set_x:N = \sexamplename
4674 }
4675 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4676   \str_clear:N \sexamplotype
4677   \str_clear:N \sexampleid
4678   \str_clear:N \sexamplename
4679   \clist_clear:N \l__stex_statements_sexample_for_clist
4680   \keys_set:nn { stex / inlineex }{ #1 }
4681 }
4682 \NewDocumentCommand \inlineex { 0{ } m } {
4683   \begingroup
4684   \stex_reactivate_macro:N \premise
4685   \stex_reactivate_macro:N \conclusion
4686   \__stex_statements_inlineex_args:n{ #1 }
4687   \str_if_empty:NF \sexampleid {
4688     \stex_ref_new_doc_target:n \sexampleid
4689   }
4690   \stex_if_smsmode:TF{
4691     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4692   }{
4693     \seq_clear:N \l_tmpa_seq
4694     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4695       \tl_if_empty:nF{ ##1 }{
4696         \stex_get_symbol:n { ##1 }
4697         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4698           \l_stex_get_symbol_uri_str
4699         }
4700       }
4701     }
4702     \exp_args:Nnx
4703     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{
4704       \str_if_empty:NF \sexamplotype {
4705         \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4706       }
4707       #2
4708       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4709     }
4710   }
4711   \endgroup
4712   \stex_smsmode_do:
4713 }
```

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

33.4 Logical Paragraphs

sparagraph

```

4714 \keys_define:nn { stex / sparagraph } {
4715   id       .str_set_x:N = \sparagraphid ,
4716   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4717   type     .str_set_x:N = \sparagraphtype ,
4718   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4719   from     .tl_set:N    = \sparagraphfrom ,
4720   to       .tl_set:N    = \sparagraphto ,
4721   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4722   name     .str_set:N   = \sparagraphname
4723 }
4724
4725 \cs_new_protected:Nn \stex_sparagraph_args:n {
4726   \tl_clear:N \l_stex_sparagraph_title_tl
4727   \tl_clear:N \sparagraphfrom
4728   \tl_clear:N \sparagraphto
4729   \tl_clear:N \l_stex_sparagraph_start_tl
4730   \str_clear:N \sparagraphid
4731   \str_clear:N \sparagraphtype
4732   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4733   \str_clear:N \sparagraphname
4734   \keys_set:nn { stex / sparagraph } { #1 }
4735 }
4736 \newif\if@in@omtext\@in@omtextfalse
4737
4738 \NewDocumentEnvironment {sparagraph} { 0{} } {
4739   \stex_sparagraph_args:n { #1 }
4740   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4741     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4742   }{
4743     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4744   }
4745   \@in@omtexttrue
4746   \stex_if_smsmode:F {
4747     \seq_clear:N \l_tmpa_seq
4748     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4749       \tl_if_empty:NF{ ##1 }{
4750         \stex_get_symbol:n { ##1 }
4751         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4752           \l_stex_get_symbol_uri_str
4753         }
4754       }
4755     }
4756     \exp_args:Nnnx
4757     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4758     \str_if_empty:NF \sparagraphtype {
4759       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4760     }
4761     \str_if_empty:NF \sparagraphfrom {
4762       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4763     }
4764     \str_if_empty:NF \sparagraphto {

```

```

4765     \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4766   }
4767   \clist_set:No \l_tmpa_clist \sparagraphtype
4768   \tl_clear:N \l_tmpa_tl
4769   \clist_map_inline:Nn \sparagraphtype {
4770     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4771       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4772     }
4773   }
4774   \tl_if_empty:NTF \l_tmpa_tl {
4775     \__stex_statements_sparagraph_start:
4776   }{
4777     \l_tmpa_tl
4778   }
4779 }
4780 \clist_set:No \l_tmpa_clist \sparagraphtype
4781 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4782 {
4783   \stex_reactivate_macro:N \definiendum
4784   \stex_reactivate_macro:N \definame
4785   \stex_reactivate_macro:N \Definame
4786   \stex_reactivate_macro:N \premise
4787   \stex_reactivate_macro:N \definiens
4788 }
4789 \str_if_empty:NTF \sparagraphid {
4790   \str_if_empty:NTF \sparagraphname {
4791     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4792       \stex_ref_new_doc_target:n {}
4793     }
4794   } {
4795     \stex_ref_new_doc_target:n {}
4796   }
4797 } {
4798   \stex_ref_new_doc_target:n \sparagraphid
4799 }
4800 \exp_args:NNx
4801 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4802   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4803     \tl_if_empty:nF{ ##1 }{
4804       \stex_get_symbol:n { ##1 }
4805       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4806     }
4807   }
4808 }
4809 \stex_smsmode_do:
4810 \ignorespacesandpars
4811 }{
4812   \str_if_empty:NF \sparagraphname {
4813     \stex_symdecl_do:nn{}{\sparagraphname}
4814     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4815   }
4816   \stex_if_smsmode:F {
4817     \clist_set:No \l_tmpa_clist \sparagraphtype
4818     \tl_clear:N \l_tmpa_tl

```



```

4819 \clist_map_inline:Nn \l_tmpa_clist {
4820   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4821     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4822   }
4823 }
4824 \tl_if_empty:NTF \l_tmpa_tl {
4825   \__stex_statements_sparagraph_end:
4826 }{
4827   \l_tmpa_tl
4828 }
4829 \end{stex_annotate_env}
4830 }
4831 }

```

\stexpatchparagraph

```

4832
4833 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4834   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4835     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4836       \titleemph{\l_stex_sparagraph_title_tl}:~
4837     }
4838   }{
4839     \titleemph{\l_stex_sparagraph_start_tl}~
4840   }
4841 }
4842 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4843
4844 \newcommand\stexpatchparagraph[3]{} {
4845   \str_set:Nx \l_tmpa_str{ #1 }
4846   \str_if_empty:NTF \l_tmpa_str {
4847     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4848     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4849   }{
4850     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4851     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4852   }
4853 }
4854
4855 \keys_define:nn { stex / inlinepara } {
4856   id      .str_set_x:N = \sparagraphid ,
4857   type    .str_set_x:N = \sparagraphtype ,
4858   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4859   from    .tl_set:N    = \sparagraphfrom ,
4860   to      .tl_set:N    = \sparagraphto ,
4861   name    .str_set:N   = \sparagraphname
4862 }
4863 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4864   \tl_clear:N \sparagraphfrom
4865   \tl_clear:N \sparagraphto
4866   \str_clear:N \sparagraphid
4867   \str_clear:N \sparagraphtype
4868   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4869   \str_clear:N \sparagraphname
4870   \keys_set:nn { stex / inlinepara }{ #1 }

```

```

4871 }
4872 \NewDocumentCommand \inlinepara { 0{} m } {
4873   \beginingroup
4874   \__stex_statements_inlinepara_args:n{ #1 }
4875   \clist_set:No \l_tmpa_clist \sparagraphtype
4876   \str_if_empty:NTF \sparaagraphid {
4877     \str_if_empty:NTF \sparaagraphname {
4878       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4879         \stex_ref_new_doc_target:n {}
4880       }
4881     } {
4882       \stex_ref_new_doc_target:n {}
4883     }
4884   } {
4885     \stex_ref_new_doc_target:n \sparaagraphid
4886   }
4887   \stex_if_smsmode:TF{
4888     \str_if_empty:NF \sparaagraphname {
4889       \stex_symdecl_do:nn{}{\sparaagraphname}
4890       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
4891     }
4892   }{
4893     \seq_clear:N \l_tmpa_seq
4894     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
4895       \tl_if_empty:nF{ ##1 }{
4896         \stex_get_symbol:n { ##1 }
4897         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4898           \l_stex_get_symbol_uri_str
4899         }
4900       }
4901     }
4902     \exp_args:Nnx
4903     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4904       \str_if_empty:NF \sparaagraphtype {
4905         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
4906       }
4907       \str_if_empty:NF \sparaagraphfrom {
4908         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
4909       }
4910       \str_if_empty:NF \sparaagraphto {
4911         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
4912       }
4913       \str_if_empty:NF \sparaagraphname {
4914         \stex_symdecl_do:nn{}{\sparaagraphname}
4915         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
4916       }
4917       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4918         \clist_map_inline:Nn \l_tmpa_seq {
4919           \stex_ref_new_sym_target:n {##1}
4920         }
4921       }
4922     } #2
4923   }
4924 }

```

```

4925 \endgroup
4926 \stex_smsmode_do:
4927 }
4928
(End definition for \stexpatchparagraph. This function is documented on page ??.)
4929 </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

```
4930 <*package>
4931 <@@=stex_sproof>
4932
4933 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4934
```

34.2 Proofs

We first define some keys for the proof environment.

```
4935 \keys_define:nn { stex / spf } {
4936   id          .str_set_x:N = \spfid,
4937   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
4938   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
4939   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
4940   type        .str_set_x:N = \spftype,
4941   title       .tl_set:N    = \spftitle,
4942   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
4943   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
4944   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
4945 }
4946 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4947   \str_clear:N \spfid
4948   \tl_clear:N \l__stex_sproof_spf_for_tl
4949   \tl_clear:N \l__stex_sproof_spf_from_tl
4950   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4951   \str_clear:N \spftype
4952   \tl_clear:N \spftitle
4953   \tl_clear:N \l__stex_sproof_spf_continues_tl
4954   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4955 \tl_clear:N \l__stex_sproof_spf_method_tl
4956 \bool_set_false:N \l__stex_sproof_inc_counter_bool
4957 \keys_set:nn { stex / spf }{ #1 }
4958 }

```

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

```

4959 \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.

```

4960 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
4961 \cs_new_protected:Npn \sproofnumber {
4962   \int_set:Nn \l_tmpa_int {1}
4963   \bool_while_do:nn {
4964     \int_compare_p:nNn {
4965       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
4966     } > 0
4967   }{
4968     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
4969     \int_incr:N \l_tmpa_int
4970   }
4971 }
4972 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
4973   \int_set:Nn \l_tmpa_int {1}
4974   \bool_while_do:nn {
4975     \int_compare_p:nNn {
4976       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
4977     } > 0
4978   }{
4979     \int_incr:N \l_tmpa_int
4980   }
4981   \int_compare:nNnF \l_tmpa_int = 1 {
4982     \int_decr:N \l_tmpa_int
4983   }
4984   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
4985     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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


```

5030 }
5031 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5032   \input{sproof-finnish.ldf}
5033 }
5034 \clist_if_in:NnT \l_tmpa_clist {french}{
5035   \input{sproof-french.ldf}
5036 }
5037 \clist_if_in:NnT \l_tmpa_clist {russian}{
5038   \input{sproof-russian.ldf}
5039 }
5040 \makeatother
5041 }{}
5042 }

```

spfsketch

```

5043 \newcommand\spfsketch[2] [] {
5044   \beginingroup
5045   \let \premise \stex_proof_premise:
5046   \__stex_sproof_spf_args:n{#1}
5047   \stex_if_smsmode:TF {
5048     \str_if_empty:NF \spfid {
5049       \stex_ref_new_doc_target:n \spfid
5050     }
5051   }{
5052     \seq_clear:N \l_tmpa_seq
5053     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5054       \tl_if_empty:nF{ ##1 }{
5055         \stex_get_symbol:n { ##1 }
5056         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5057           \l_stex_get_symbol_uri_str
5058         }
5059       }
5060     }
5061     \exp_args:Nnx
5062     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5063       \str_if_empty:NF \spftype {
5064         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5065       }
5066       \clist_set:No \l_tmpa_clist \spftype
5067       \tl_set:Nn \l_tmpa_tl {
5068         \titleemph{
5069           \tl_if_empty:NTF \spftitle {
5070             \spf@proofsketch@kw
5071           }{
5072             \spftitle
5073           }
5074         }::~
5075       }
5076       \clist_map_inline:Nn \l_tmpa_clist {
5077         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5078           \tl_clear:N \l_tmpa_tl
5079         }
5080       }
5081       \str_if_empty:NF \spfid {

```

```

5082         \stex_ref_new_doc_target:n \spfid
5083     }
5084     \l_tmpa_tl #2 \sproofend
5085 }
5086 }
5087 \endgroup
5088 \stex_smsmode_do:
5089 }
5090

```

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

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

```

5091 \newenvironment{spfeq}[2][]{
5092   \__stex_sproof_spf_args:n{#1}
5093   \let \premise \stex_proof_premise:
5094   \stex_if_smsmode:TF {
5095     \str_if_empty:NF \spfid {
5096       \stex_ref_new_doc_target:n \spfid
5097     }
5098   }{
5099     \seq_clear:N \l_tmpa_seq
5100     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5101       \tl_if_empty:NF{ ##1 }{
5102         \stex_get_symbol:n { ##1 }
5103         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5104           \l_stex_get_symbol_uri_str
5105         }
5106       }
5107     }
5108     \exp_args:Nnnx
5109     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5110     \str_if_empty:NF \spftype {
5111       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5112     }
5113
5114     \clist_set:No \l_tmpa_clist \spftype
5115     \tl_clear:N \l_tmpa_tl
5116     \clist_map_inline:Nn \l_tmpa_clist {
5117       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5118         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5119       }
5120       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5121         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5122       }
5123     }
5124     \tl_if_empty:NTF \l_tmpa_tl {
5125       \__stex_sproof_spfeq_start:
5126     }{
5127       \l_tmpa_tl
5128     }{-#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


```

5129 \str_if_empty:NF \spfid {
5130 \stex_ref_new_doc_target:n \spfid
5131 }
5132 \begin{displaymath}\begin{array}{rcll}
5133 }
5134 \stex_smsmode_do:
5135 }{
5136 \stex_if_smsmode:F {
5137 \end{array}\end{displaymath}
5138 \clist_set:No \l_tmpa_clist \spftype
5139 \tl_clear:N \l_tmpa_tl
5140 \clist_map_inline:Nn \l_tmpa_clist {
5141 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5142 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5143 }
5144 }
5145 \tl_if_empty:NTF \l_tmpa_tl {
5146 \__stex_sproof_spfeq_end:
5147 }{
5148 \l_tmpa_tl
5149 }
5150 \end{stex_annotate_env}
5151 }
5152 }
5153
5154 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5155 \titleemph{
5156 \tl_if_empty:NTF \spftitle {
5157 \spf@proof@kw
5158 }{
5159 \spftitle
5160 }
5161 }:
5162 }
5163 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5164
5165 \newcommand\stexpatchspfeq[3] [] {
5166 \str_set:Nx \l_tmpa_str{ #1 }
5167 \str_if_empty:NTF \l_tmpa_str {
5168 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5169 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5170 }{
5171 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5172 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5173 }
5174 }
5175

```

(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.

```

5176 \newenvironment{sproof}[2] []{

```

```

5177 \let \premise \stex_proof_premise:
5178 \intarray_gzero:N \l__stex_sproof_counter_intarray
5179 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5180 \__stex_sproof_spf_args:n{#1}
5181 \stex_if_smsmode:TF {
5182   \str_if_empty:NF \spfid {
5183     \stex_ref_new_doc_target:n \spfid
5184   }
5185 }{
5186   \seq_clear:N \l_tmpa_seq
5187   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5188     \tl_if_empty:NF{ ##1 }{
5189       \stex_get_symbol:n { ##1 }
5190       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5191         \l_stex_get_symbol_uri_str
5192       }
5193     }
5194   }
5195   \exp_args:Nnnx
5196   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5197   \str_if_empty:NF \spftype {
5198     \stex_annotate_invisible:nnn{type}{\spftype}{}
5199   }
5200
5201   \clist_set:No \l_tmpa_clist \spftype
5202   \tl_clear:N \l_tmpa_tl
5203   \clist_map_inline:Nn \l_tmpa_clist {
5204     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5205       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5206     }
5207     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5208       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5209     }
5210   }
5211   \tl_if_empty:NTF \l_tmpa_tl {
5212     \__stex_sproof_sproof_start:
5213   }{
5214     \l_tmpa_tl
5215   }{~#2}
5216   \str_if_empty:NF \spfid {
5217     \stex_ref_new_doc_target:n \spfid
5218   }
5219   \begin{description}
5220 }
5221 \stex_smsmode_do:
5222 }{
5223   \stex_if_smsmode:F{
5224     \end{description}
5225     \clist_set:No \l_tmpa_clist \spftype
5226     \tl_clear:N \l_tmpa_tl
5227     \clist_map_inline:Nn \l_tmpa_clist {
5228       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5229         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5230       }

```

```

5231 }
5232 \tl_if_empty:NTF \l_tmpa_tl {
5233   \__stex_sproof_sproof_end:
5234 }{
5235   \l_tmpa_tl
5236 }
5237 \end{stex_annotate_env}
5238 }
5239 }
5240
5241 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5242   \par\noindent\titleemph{
5243     \tl_if_empty:NTF \spftype {
5244       \spf@proof@kw
5245     }{
5246       \spftype
5247     }
5248   }:
5249 }
5250 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5251
5252 \newcommand\stexpatchsproof[3] [] {
5253   \str_set:Nx \l_tmpa_str{ #1 }
5254   \str_if_empty:NTF \l_tmpa_str {
5255     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5256     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5257   }{
5258     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5259     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5260   }
5261 }

```

`\spfidea`

```

5262 \newcommand\spfidea[2] []{
5263   \__stex_sproof_spf_args:n{#1}
5264   \titleemph{
5265     \tl_if_empty:NTF \spftype {Proof~Idea}{
5266       \spftype
5267     }:
5268   }~#2
5269   \sproofend
5270 }

```

(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`

```

5271 \newenvironment{spfstep}[1] []{
5272   \__stex_sproof_spf_args:n{#1}
5273   \stex_if_smsmode:TF {

```

```

5274 \str_if_empty:NF \spfid {
5275 \stex_ref_new_doc_target:n \spfid
5276 }
5277 }{
5278 \in@contexttrue
5279 \seq_clear:N \l_tmpa_seq
5280 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5281 \tl_if_empty:NF{ ##1 }{
5282 \stex_get_symbol:n { ##1 }
5283 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5284 \l_stex_get_symbol_uri_str
5285 }
5286 }
5287 }
5288 \exp_args:Nnnx
5289 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5290 \str_if_empty:NF \spftype {
5291 \stex_annotate_invisible:nnn{type}{\spftype}{}
5292 }
5293 \clist_set:No \l_tmpa_clist \spftype
5294 \tl_set:Nn \l_tmpa_tl {
5295 \item[\sproofnumber]
5296 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5297 }
5298 \clist_map_inline:Nn \l_tmpa_clist {
5299 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5300 \tl_clear:N \l_tmpa_tl
5301 }
5302 }
5303 \l_tmpa_tl
5304 \tl_if_empty:NF \spftitle {
5305 {(\titleemph{\spftitle})\enspace}
5306 }
5307 \str_if_empty:NF \spfid {
5308 \stex_ref_new_doc_target:n \spfid
5309 }
5310 }
5311 \stex_smsmode_do:
5312 \ignorespacesandpars
5313 }{
5314 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5315 \__stex_sproof_inc_counter:
5316 }
5317 \stex_if_smsmode:F {
5318 \end{stex_annotate_env}
5319 }
5320 }

```

sproofcomment

```

5321 \newenvironment{sproofcomment}[1][]{
5322 \__stex_sproof_spf_args:n{#1}
5323 \clist_set:No \l_tmpa_clist \spftype
5324 \tl_set:Nn \l_tmpa_tl {
5325 \item[\sproofnumber]

```

```

5326 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5327 }
5328 \clist_map_inline:Nn \l_tmpa_clist {
5329 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5330 \tl_clear:N \l_tmpa_tl
5331 }
5332 }
5333 \l_tmpa_tl
5334 }{
5335 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5336 \__stex_sproof_inc_counter:
5337 }
5338 }

```

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.

```

5339 \newenvironment{subproof}[2][]{
5340 \__stex_sproof_spf_args:n{#1}
5341 \stex_if_smsmode:TF{
5342 \str_if_empty:NF \spfid {
5343 \stex_ref_new_doc_target:n \spfid
5344 }
5345 }{
5346 \seq_clear:N \l_tmpa_seq
5347 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5348 \tl_if_empty:nF{ ##1 }{
5349 \stex_get_symbol:n { ##1 }
5350 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5351 \l_stex_get_symbol_uri_str
5352 }
5353 }
5354 }
5355 \exp_args:Nnnx
5356 \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5357 \str_if_empty:NF \spftype {
5358 \stex_annotate_invisible:nnn{type}{\spftype}{}}
5359 }
5360
5361 \clist_set:No \l_tmpa_clist \spftype
5362 \tl_set:Nn \l_tmpa_tl {
5363 \item[\sproofnumber]
5364 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5365 }
5366 \clist_map_inline:Nn \l_tmpa_clist {
5367 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5368 \tl_clear:N \l_tmpa_tl
5369 }
5370 }
5371 \l_tmpa_tl
5372 \tl_if_empty:NF \spftitle {
5373 {(\titleemph{\spftitle})\enspace}
5374 }

```

```

5375     {~#2}
5376     \str_if_empty:NF \spfid {
5377       \stex_ref_new_doc_target:n \spfid
5378     }
5379   }
5380   \__stex_sproof_add_counter:
5381   \stex_smsmode_do:
5382 }{
5383   \__stex_sproof_remove_counter:
5384   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5385     \__stex_sproof_inc_counter:
5386   }
5387   \stex_if_smsmode:F{
5388     \end{stex_annotate_env}
5389   }
5390 }

```

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

```

5391 \newenvironment{spfcases}[2][]{
5392   \tl_if_empty:nTF{#1}{
5393     \begin{subproof}[method=by-cases]{#2}
5394   }{
5395     \begin{subproof}[#1,method=by-cases]{#2}
5396   }
5397 }{
5398   \end{subproof}
5399 }

```

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

```

5400 \newenvironment{spfcase}[2][]{
5401   \__stex_sproof_spf_args:n{#1}
5402   \stex_if_smsmode:TF {
5403     \str_if_empty:NF \spfid {
5404       \stex_ref_new_doc_target:n \spfid
5405     }
5406   }{
5407     \seq_clear:N \l_tmpa_seq
5408     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5409       \tl_if_empty:nF{ ##1 }{
5410         \stex_get_symbol:n { ##1 }
5411         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5412           \l_stex_get_symbol_uri_str
5413         }
5414       }
5415     }
5416     \exp_args:Nnnx
5417     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5418     \str_if_empty:NF \spftype {
5419       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5420   }
5421   \clist_set:Nn \l_tmpa_clist \spftype
5422   \tl_set:Nn \l_tmpa_tl {
5423     \item[\sproofnumber]

```

```

5424     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5425   }
5426   \clist_map_inline:Nn \l_tmpa_clist {
5427     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5428       \tl_clear:N \l_tmpa_tl
5429     }
5430   }
5431   \l_tmpa_tl
5432   \tl_if_empty:nF{#2}{
5433     \titleemph{#2}:~
5434   }
5435 }
5436 \__stex_sproof_add_counter:
5437 \stex_smsmode_do:
5438 ){
5439   \__stex_sproof_remove_counter:
5440   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5441     \__stex_sproof_inc_counter:
5442   }
5443   \stex_if_smsmode:F{
5444     \clist_set:No \l_tmpa_clist \spftype
5445     \tl_set:Nn \l_tmpa_tl{\sproofend}
5446     \clist_map_inline:Nn \l_tmpa_clist {
5447       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5448         \tl_clear:N \l_tmpa_tl
5449       }
5450     }
5451     \l_tmpa_tl
5452     \end{stex_annotate_env}
5453   }
5454 }

```

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

```

5455 \newcommand\spfcasesketch[3][]{
5456   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5457 }

```

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.

```

5458 \keys_define:nn { stex / just }{
5459   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5460   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5461   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5462   args    .tl_set:N    = \l__stex_sproof_just_args_tl
5463 }

```

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

```
5464 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5465 \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.

```
5466 \newcommand\justarg[2] [] {#2}
```

```
5467 \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

```
5468 <*package>
5469
5470 %%%%%%%%%%% others.dtx %%%%%%%%%%%
5471
5472 <@@=stex_others>
    Warnings and error messages
5473 % None

\MSC Math subject classifier

5474 \NewDocumentCommand \MSC {m} {
5475 % TODO
5476 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5477 \@ifpackageloaded{tikzinput}{
5478 \RequirePackage{stex-tikzinput}
5479 }{}
5480 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
5481 <*package>
5482 <@@=stex_modules>
5483
5484 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5485
5486 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5487 \begingroup
5488 \stex_module_setup:nn{
5489   ns=\c_stex_metatheory_ns_str,
5490   meta=NONE
5491 }{Metatheory}
5492 \stex_reactivate_macro:N \symdecl
5493 \stex_reactivate_macro:N \notation
5494 \stex_reactivate_macro:N \symdef
5495 \ExplSyntaxOff
5496 \csname stex_suppress_html:n\endcsname{
5497   % is-a (a:A, a \in A, a is an A, etc.)
5498   \symdecl{isa}[args=ai]
5499   \notation{isa}[typed]{#1 \comp{:} #2}{##1 \comp, ##2}
5500   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5501   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5502
5503   % bind (\forall, \Pi, \lambda etc.)
5504   \symdecl{bind}[args=Bi]
5505   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
5506   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5507   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
5508
5509   % dummy variable
5510   \symdecl{dummyvar}
5511   \notation{dummyvar}[underscore]{\comp\_}
5512   \notation{dummyvar}[dot]{\comp\cdot}
5513   \notation{dummyvar}[dash]{\comp{\rm --}}
5514
5515   %fromto (function space, Hom-set, implication etc.)
```

```

5516 \symdecl{fromto}[args=ai]
5517 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5518 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5519
5520 % mapto (lambda etc.)
5521 \symdecl{mapto}[args=Bi]
5522 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5523 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5524 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5525
5526 % function/operator application
5527 \symdecl{apply}[args=ia]
5528 \notation{apply}[prec=0;0x\infpref,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5529 \notation{apply}[prec=0;0x\infpref,lambda]{#1 \; #2 }{##1 \; ; ##2}
5530
5531 % ‘type’ of all collections (sets, classes, types, kinds)
5532 \symdecl{collection}
5533 \notation{collection}[U]{\comp{\mathcal{U}}}
5534 \notation{collection}[set]{\comp{\textsf{Set}}}
5535
5536 % collection of propositions/booleans/truth values
5537 \symdecl{prop}[name=proposition]
5538 \notation{prop}[prop]{\comp{\rm prop}}
5539 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5540
5541 % sequences
5542 \symdecl{seqtype}[args=1]
5543 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5544
5545 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5546 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
5547
5548 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
5549 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
5550 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
5551
5552 % letin (‘let’, local definitions, variable substitution)
5553 \symdecl{letin}[args=bii]
5554 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;.\; \comp{\rm in}}\;#3}
5555 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5556 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5557
5558 % structures
5559 \symdecl*{module-type}[args=1]
5560 \notation{module-type}{\mathtt{MOD} #1}
5561 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5562 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5563
5564 }
5565 \ExplSyntaxOn
5566 \stex_add_to_current_module:n{
5567   \let\appa\apply
5568   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5569   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5570 \def\livar{\csname sequence-index\endcsname[li]}
5571 \def\uivar{\csname sequence-index\endcsname[ui]}
5572 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5573 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5574 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5575 }
5576 \__stex_modules_end_module:
5577 \endgroup
5578 </package>

```

Chapter 37

Tikzinput Implementation

```
5579 <*package>
5580
5581 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5582
5583 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5584 \RequirePackage{l3keys2e}
5585
5586 \keys_define:nn { tikzinput } {
5587   image .bool_set:N = \c_tikzinput_image_bool,
5588   image .default:n = false ,
5589   unknown .code:n = {}
5590 }
5591
5592 \ProcessKeysOptions { tikzinput }
5593
5594 \bool_if:NTF \c_tikzinput_image_bool {
5595   \RequirePackage{graphicx}
5596
5597   \providecommand\usetikzlibrary[]{}
5598   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5599 }{
5600   \RequirePackage{tikz}
5601   \RequirePackage{standalone}
5602
5603   \newcommand \tikzinput [2] [] {
5604     \setkeys{Gin}{#1}
5605     \ifx \Gin@ewidth \Gin@exclamation
5606       \ifx \Gin@eheight \Gin@exclamation
5607         \input { #2 }
5608       \else
5609         \resizebox{!}{ \Gin@eheight }{
5610           \input { #2 }
5611         }
5612       \fi
5613     \else
5614       \ifx \Gin@eheight \Gin@exclamation
5615         \resizebox{ \Gin@ewidth }{!}{
5616           \input { #2 }
```

```

5617     }
5618     \else
5619         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5620             \input { #2 }
5621         }
5622     \fi
5623 \fi
5624 }
5625 }
5626
5627 \newcommand \ctikzinput [2] [] {
5628     \begin{center}
5629         \tikzinput [ #1 ] { #2 }
5630     \end{center}
5631 }
5632
5633 \@ifpackageloaded{stex}{
5634     \RequirePackage{stex-tikzinput}
5635 }{}
5636
5637 </package>
5638 <*stex>
5639 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5640 \RequirePackage{stex}
5641 \RequirePackage{tikzinput}
5642
5643 \newcommand\mhtikzinput [2] [] {%
5644     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5645     \stex_in_repository:nn\Gin@mhrepos{
5646         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5647     }
5648 }
5649 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5650 </stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

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

```
5651 \*cls)
5652 \@@=document_structure)
5653 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5654 \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`

```
5655 \keys_define:nn{ document-structure / pkg }{
5656   class      .str_set_x:N = \c_document_structure_class_str,
5657   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5658   report     .code:n      = {
5659     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5660     \str_set:Nn \c_document_structure_class_str {report}
5661   },
5662   book       .code:n      = {
5663     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5664     \str_set:Nn \c_document_structure_class_str {book}
5665   },
5666   bookpart   .code:n      = {
5667     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5668     \str_set:Nn \c_document_structure_class_str {book}
5669     \str_set:Nn \c_document_structure_topsect_str {chapter}
5670   },
```

```

5671 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5672 unknown     .code:n      = {
5673   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5674 }
5675 }
5676 \ProcessKeysOptions{ document-structure / pkg }
5677 \str_if_empty:NT \c_document_structure_class_str {
5678   \str_set:Nn \c_document_structure_class_str {article}
5679 }
5680 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5681   {\c_document_structure_class_str}
5682

```

38.3 Beefing up the document environment

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

```

5683 \RequirePackage{document-structure}
5684 \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

```

5685 \keys_define:nn { document-structure / document }{
5686   id .str_set_x:N = \c_document_structure_document_id_str
5687 }
5688 \let\__document_structure_orig_document=\document
5689 \renewcommand{\document}[1][]{
5690   \keys_set:nn{ document-structure / document }{ #1 }
5691   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5692   \__document_structure_orig_document
5693 }

```

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

```

5694 }
5695 \</cls>

```

38.4 Implementation: document-structure Package

```

5696 \<*package>
5697 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
5698 \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


```

5699
5700 \keys_define:nn{ document-structure / pkg }{
5701   class      .str_set_x:N = \c_document_structure_class_str,
5702   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5703   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5704 }
5705 \ProcessKeysOptions{ document-structure / pkg }
5706 \str_if_empty:NT \c_document_structure_class_str {
5707   \str_set:Nn \c_document_structure_class_str {article}
5708 }
5709 \str_if_empty:NT \c_document_structure_topsect_str {
5710   \str_set:Nn \c_document_structure_topsect_str {section}
5711 }

```

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

```

5712 \RequirePackage{xspace}
5713 \RequirePackage{comment}
5714 \AddToHook{begindocument}{
5715   \ltx@ifpackageloaded{babel}{
5716     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5717     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5718       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
5719     }
5720   }{}
5721 }

```

`\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.

```

5722 \int_new:N \l_document_structure_section_level_int
5723 \str_case:VnF \c_document_structure_topsect_str {
5724   {part}}{
5725     \int_set:Nn \l_document_structure_section_level_int {0}
5726   }
5727   {chapter}{
5728     \int_set:Nn \l_document_structure_section_level_int {1}
5729   }
5730 }{
5731   \str_case:VnF \c_document_structure_class_str {
5732     {book}}{
5733       \int_set:Nn \l_document_structure_section_level_int {0}
5734     }
5735     {report}}{
5736       \int_set:Nn \l_document_structure_section_level_int {0}
5737     }
5738   }{
5739     \int_set:Nn \l_document_structure_section_level_int {2}
5740   }
5741 }

```

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

```
5742 \def\current@section@level{document}%
5743 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5744 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5745 \cs_new_protected:Npn \skipomgroup {
5746   \ifcase\l_document_structure_section_level_int
5747   \or\stepcounter{part}
5748   \or\stepcounter{chapter}
5749   \or\stepcounter{section}
5750   \or\stepcounter{subsection}
5751   \or\stepcounter{subsubsection}
5752   \or\stepcounter{paragraph}
5753   \or\stepcounter{subparagraph}
5754   \fi
5755 }
```

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

`blindfragment`

```
5756 \newcommand\at@begin@blindomgroup[1]{%
5757 \newenvironment{blindfragment}
5758 {
5759   \int_incr:N\l_document_structure_section_level_int
5760   \at@begin@blindomgroup\l_document_structure_section_level_int
5761 }{}}
```

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

```
5762 \newcommand\omgroup@nonum[2]{
5763   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5764   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5765 }
```

(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.

```
5766 \newcommand\omgroup@num[2]{
```

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

```

5767 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5768   \@nameuse{#1}{#2}
5769 }{
5770   \cs_if_exist:NTF\rdfmata@sectioning{
5771     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5772   }{
5773     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5774   }
5775 }
5776 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5777 }

```

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

sfragment

```

5778 \keys_define:nn { document-structure / omgroup }{
5779   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5780   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5781   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
5782   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5783   srccite      .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5784   type         .tl_set:N    = \l__document_structure_omgroup_type_tl,
5785   short        .tl_set:N    = \l__document_structure_omgroup_short_tl,
5786   display      .tl_set:N    = \l__document_structure_omgroup_display_tl,
5787   intro        .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5788   loadmodules  .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5789 }
5790 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
5791   \str_clear:N \l__document_structure_omgroup_id_str
5792   \str_clear:N \l__document_structure_omgroup_date_str
5793   \clist_clear:N \l__document_structure_omgroup_creators_clist
5794   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5795   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5796   \tl_clear:N \l__document_structure_omgroup_type_tl
5797   \tl_clear:N \l__document_structure_omgroup_short_tl
5798   \tl_clear:N \l__document_structure_omgroup_display_tl
5799   \tl_clear:N \l__document_structure_omgroup_intro_tl
5800   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5801   \keys_set:nn { document-structure / omgroup } { #1 }
5802 }

```

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.

```

5803 \newif\if@mainmatter\@mainmattertrue
5804 \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.

```

5805 \keys_define:nn { document-structure / sectioning }{
5806   name      .str_set_x:N = \l__document_structure_sect_name_str ,
5807   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
5808   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
5809   clear     .default:n   = {true} ,
5810   num      .bool_set:N  = \l__document_structure_sect_num_bool  ,

```

```

5811   num      .default:n      = {true}
5812 }
5813 \cs_new_protected:Nn \__document_structure_sect_args:n {
5814   \str_clear:N \l__document_structure_sect_name_str
5815   \str_clear:N \l__document_structure_sect_ref_str
5816   \bool_set_false:N \l__document_structure_sect_clear_bool
5817   \bool_set_false:N \l__document_structure_sect_num_bool
5818   \keys_set:nn { document-structure / sectioning } { #1 }
5819 }
5820 \newcommand\omdoc@sectioning[3][]{
5821   \__document_structure_sect_args:n {#1}
5822   \let\omdoc@sect@name\l__document_structure_sect_name_str
5823   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5824   \if@mainmatter% numbering not overridden by frontmatter, etc.
5825     \bool_if:NTF \l__document_structure_sect_num_bool {
5826       \omgroup@num{#2}{#3}
5827     }{
5828       \omgroup@nonum{#2}{#3}
5829     }
5830     \def\current@section@level{\omdoc@sect@name}
5831   \else
5832     \omgroup@nonum{#2}{#3}
5833   \fi
5834 }% 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.

```

5835 \newcommand\omgroup@redefine@addtocontents[1]{%
5836 %\edef\__document_structureimport{#1}%
5837 %\@for\@I:=\__document_structureimport\do{%
5838 %\edef\@path{\csname module@\@I @path\endcsname}%
5839 %\@ifundefined{tf@toc}\relax%
5840 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5841 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5842 %\def\addcontentsline##1##2##3{%
5843 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5844 %\else% hyperref.sty not loaded
5845 %\def\addcontentsline##1##2##3{%
5846 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5847 %\fi
5848 }% 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.

```

5849 \newenvironment{sfragment}[2][]{% keys, title
5850 {
5851   \__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.

```

5852   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5853     \omgroup@redefine@addtocontents{
5854       \@ifundefined{module@id}\used@modules%

```

```

5855     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5856   }
5857 }

```

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

```

5858 \int_incr:N\l_document_structure_section_level_int
5859 \ifcase\l_document_structure_section_level_int
5860   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5861   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5862   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5863   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5864   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5865   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5866   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5867 \fi
5868 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5869 \str_if_empty:NF \l__document_structure_omgroup_id_str {
5870   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5871 }
5872 }% for customization
5873 {}

```

and finally, we localize the sections

```

5874 \newcommand\omdoc@part@kw{Part}
5875 \newcommand\omdoc@chapter@kw{Chapter}
5876 \newcommand\omdoc@section@kw{Section}
5877 \newcommand\omdoc@subsection@kw{Subsection}
5878 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5879 \newcommand\omdoc@paragraph@kw{paragraph}
5880 \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`

```

5881 \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).

```

5882 \cs_if_exist:NTF\frontmatter{
5883   \let\__document_structure_orig_frontmatter\frontmatter
5884   \let\frontmatter\relax
5885 }{
5886   \tl_set:Nn\__document_structure_orig_frontmatter{
5887     \clearpage
5888     \@mainmatterfalse
5889     \pagenumbering{roman}

```

```

5890 }
5891 }
5892 \cs_if_exist:NTF\backmatter{
5893   \let\__document_structure_orig_backmatter\backmatter
5894   \let\backmatter\relax
5895 }{
5896   \tl_set:Nn\__document_structure_orig_backmatter{
5897     \clearpage
5898     \@mainmatterfalse
5899     \pagenumbering{roman}
5900   }
5901 }

```

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.

```

5902 \newenvironment{frontmatter}{
5903   \__document_structure_orig_frontmatter
5904 }{
5905   \cs_if_exist:NTF\mainmatter{
5906     \mainmatter
5907   }{
5908     \clearpage
5909     \@mainmattertrue
5910     \pagenumbering{arabic}
5911   }
5912 }

```

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

```

5913 \newenvironment{backmatter}{
5914   \__document_structure_orig_backmatter
5915 }{
5916   \cs_if_exist:NTF\mainmatter{
5917     \mainmatter
5918   }{
5919     \clearpage
5920     \@mainmattertrue
5921     \pagenumbering{arabic}
5922   }
5923 }

```

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

```

5924 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5925 \def \c__document_structure_document_str{document}
5926 \newcommand\afterprematurestop{}
5927 \def\prematurestop@endomgroup{
5928   \unless\ifx\@currenvir\c__document_structure_document_str
5929     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5930     \expandafter\prematurestop@endomgroup

```

```

5931 \fi
5932 }
5933 \providecommand\prematurestop{
5934 \message{Stopping~sTeX~processing~prematurely}
5935 \prematurestop@endumgroup
5936 \afterprematurestop
5937 \end{document}
5938 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5939 \RequirePackage{etoolbox}
5940 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5941 \newrobustcmd\useSGvar[1]{%
5942 \@ifundefined{sTeX@Gvar@#1}
5943 {\PackageError{document-structure}
5944 {The sTeX Global variable #1 is undefined}
5945 {set it with \protect\setSGvar}}
5946 \@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.

```

5947 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5948 \@ifundefined{sTeX@Gvar@#1}
5949 {\PackageError{document-structure}
5950 {The sTeX Global variable #1 is undefined}
5951 {set it with \protect\setSGvar}}
5952 {\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.

```
5953 \*cls)
5954 \@@=notesslides}
5955 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
5956 \RequirePackage{13keys2e}
5957
5958 \keys_define:nn{notesslides / cls}{
5959   class .code:n = {
5960     \PassOptionsToClass{\CurrentOption}{document-structure}
5961     \str_if_eq:nnT{#1}{book}{
5962       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5963     }
5964     \str_if_eq:nnT{#1}{report}{
5965       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5966     }
5967   },
5968   notes .bool_set:N = \c__notesslides_notes_bool ,
5969   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5970   unknown .code:n = {
5971     \PassOptionsToClass{\CurrentOption}{document-structure}
5972     \PassOptionsToClass{\CurrentOption}{beamer}
5973     \PassOptionsToPackage{\CurrentOption}{notesslides}
5974   }
5975 }
5976 \ProcessKeysOptions{ notesslides / cls }
5977 \bool_if:NTF \c__notesslides_notes_bool {
5978   \PassOptionsToPackage{notes=true}{notesslides}
5979 }{
5980   \PassOptionsToPackage{notes=false}{notesslides}
5981 }
5982 \</cls)
```


now we do the same for the notesslides package.

```

5983 <*package>
5984 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
5985 \RequirePackage{13keys2e}
5986
5987 \keys_define:nn{notesslides / pkg}{
5988   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5989   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5990   notes        .bool_set:N = \c__notesslides_notes_bool ,
5991   slides        .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5992   sectocframes  .bool_set:N = \c__notesslides_sectocframes_bool ,
5993   frameimages   .bool_set:N = \c__notesslides_frameimages_bool ,
5994   fiboxed       .bool_set:N = \c__notesslides_fiboxed_bool ,
5995   nopproblems   .bool_set:N = \c__notesslides_nopproblems_bool,
5996   unknown       .code:n      = {
5997     \PassOptionsToClass{\CurrentOption}{stex}
5998     \PassOptionsToClass{\CurrentOption}{tikzinput}
5999   }
6000 }
6001 \ProcessKeysOptions{ notesslides / pkg }
6002 \newif\ifnotes
6003 \bool_if:NTF \c__notesslides_notes_bool {
6004   \notesttrue
6005 }{
6006   \notesfalse
6007 }
6008

```

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

```

6009 \str_if_empty:NTF \c__notesslides_topsect_str {
6010   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6011 }{
6012   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6013 }
6014 </package>

```

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

```

6015 <*cls>
6016 \bool_if:NTF \c__notesslides_notes_bool {
6017   \LoadClass{document-structure}
6018 }{
6019   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6020   \newcounter{Item}
6021   \newcounter{paragraph}
6022   \newcounter{subparagraph}
6023   \newcounter{Hfootnote}
6024   \RequirePackage{document-structure}
6025 }

```

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

```

6026 \RequirePackage{notesslides}
6027 </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.

```

6028 <*package>
6029 \bool_if:NT \c__notesslides_notes_bool {
6030   \RequirePackage{a4wide}
6031   \RequirePackage{marginnote}
6032   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6033   \RequirePackage{mdframed}
6034   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6035   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6036 }
6037 \RequirePackage{stex-tikzinput}
6038 \RequirePackage{etoolbox}
6039 \RequirePackage{amssymb}
6040 \RequirePackage{amsmath}
6041 \RequirePackage{comment}
6042 \RequirePackage{textcomp}
6043 \RequirePackage{url}
6044 \RequirePackage{graphicx}
6045 \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

```

6046 \bool_if:NT \c__notesslides_notes_bool {
6047   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6048 }
6049
6050
6051 \NewDocumentCommand \libusetheme {0{} m} {
6052   \bool_if:NTF \c__notesslides_notes_bool {
6053     \libusepackage[#1]{beamernotestheme#2}
6054   }{
6055     \libusepackage[#1]{beamertheme#2}
6056   }
6057 }

```

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

```

6058 \newcounter{slide}
6059 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6060 \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.

```

6061 \bool_if:NTF \c__notesslides_notes_bool {
6062   \renewenvironment{note}{\ignorespaces}{}
6063 }{
6064   \excludecomment{note}
6065 }

```

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.

```

6066 \bool_if:NT \c__notesslides_notes_bool {
6067   \newlength{\slideframewidth}
6068   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6069 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6070   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6071     \bool_set_true:N #1
6072   }{
6073     \bool_set_false:N #1
6074   }
6075 }
6076 \keys_define:nn{notesslides / frame}{
6077   label .str_set_x:N = \l__notesslides_frame_label_str,
6078   allowframebreaks .code:n = {
6079     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6080   },
6081   allowdisplaybreaks .code:n = {
6082     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6083   },
6084   fragile .code:n = {
6085     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6086   },
6087   shrink .code:n = {
6088     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6089   },
6090   squeeze .code:n = {
6091     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6092   },
6093   t .code:n = {
6094     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6095   },
6096 }
6097 \cs_new_protected:Nn \__notesslides_frame_args:n {
6098   \str_clear:N \l__notesslides_frame_label_str
6099   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6100   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6101   \bool_set_true:N \l__notesslides_frame_fragile_bool
6102   \bool_set_true:N \l__notesslides_frame_shrink_bool
6103   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6104   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6105     \keys_set:nn { notesslides / frame }{ #1 }
6106   }

```

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

```

6107   \renewenvironment{frame}[1][]{
6108     \__notesslides_frame_args:n{#1}
6109     \sffamily
6110     \stepcounter{slide}
6111     \def\@currentlabel{\theslide}
6112     \str_if_empty:NF \l__notesslides_frame_label_str {
6113       \label{\l__notesslides_frame_label_str}
6114     }

```

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

```

6115     \def\itemize@level{outer}
6116     \def\itemize@outer{outer}
6117     \def\itemize@inner{inner}
6118     \renewcommand\newpage{\addtocounter{framenumber}{1}}
6119     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6120     \renewenvironment{itemize}{
6121       \ifx\itemize@level\itemize@outer
6122         \def\itemize@label{$\rhd$}
6123       \fi
6124       \ifx\itemize@level\itemize@inner
6125         \def\itemize@label{$\scriptstyle\rhd$}
6126       \fi
6127       \begin{list}
6128         {\itemize@label}
6129         {\setlength{\labelsep}{.3em}
6130          \setlength{\labelwidth}{.5em}
6131          \setlength{\leftmargin}{1.5em}
6132         }
6133       \edef\itemize@level{\itemize@inner}
6134     }{
6135       \end{list}
6136     }

```

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

```

6137     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6138   }{
6139     \medskip\miko@slidelabel\end{mdframed}
6140   }

```

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

`\frametitle`

```

6141   \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6142 }

```

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

EdN:20

`\pause`

```

20
6143 \bool_if:NT \c__notesslides_notes_bool {
6144   \newcommand\pause{}
6145 }

```

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

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

nparagraph

```
6146 \bool_if:NTF \c__notesslides_notes_bool {
6147   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6148 }{
6149   \excludecomment{nparagraph}
6150 }
```

nfragment

```
6151 \bool_if:NTF \c__notesslides_notes_bool {
6152   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6153 }{
6154   \excludecomment{nfragment}
6155 }
```

ndefinition

```
6156 \bool_if:NTF \c__notesslides_notes_bool {
6157   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6158 }{
6159   \excludecomment{ndefinition}
6160 }
```

nassertion

```
6161 \bool_if:NTF \c__notesslides_notes_bool {
6162   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6163 }{
6164   \excludecomment{nassertion}
6165 }
```

nsproof

```
6166 \bool_if:NTF \c__notesslides_notes_bool {
6167   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6168 }{
6169   \excludecomment{nsproof}
6170 }
```

nexample

```
6171 \bool_if:NTF \c__notesslides_notes_bool {
6172   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6173 }{
6174   \excludecomment{nexample}
6175 }
```

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

```
6176 \def\inputref@preskip{\smallskip}
6177 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6178 \let\orig@inputref\inputref
6179 \def\inputref{\@ifstar\ninputref\orig@inputref}
6180 \newcommand\ninputref[2][] {
6181   \bool_if:NT \c__notesslides_notes_bool {
6182     \orig@inputref[#1]{#2}
6183   }
6184 }
```

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

39.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the \TeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
6185 \newlength{\slidelogoheight}
6186
6187 \bool_if:NTF \c__notesslides_notes_bool {
6188   \setlength{\slidelogoheight}{.4cm}
6189 }{
6190   \setlength{\slidelogoheight}{1cm}
6191 }
6192 \newsavebox{\slidelogo}
6193 \sbox{\slidelogo}{\text{\TeX}}
6194 \newrobustcmd{\setslidelogo}[1]{\def\source{#1}}
6195 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6196 }
```

(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.

```
6197 \def\source{Michael Kohlhase}% customize locally
6198 \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.

```
6199 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6200 \newsavebox{\cclogo}
6201 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6202 \newif\ifcchref\cchreffalse
6203 \AtBeginDocument{
6204   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6205 }
6206 \def\licensing{
6207   \ifcchref
```

```

6208     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6209 \else
6210     {\usebox{\cclogo}}
6211 \fi
6212 }
6213 \newrobustcmd{\setlicensing}[2][]{
6214     \def\@url{#1}
6215     \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6216     \ifx\@url\@empty
6217         \def\licensing{\usebox{\cclogo}}
6218     \else
6219         \def\licensing{
6220             \ifcchref
6221             \href{#1}{\usebox{\cclogo}}
6222             \else
6223             {\usebox{\cclogo}}
6224             \fi
6225         }
6226     \fi
6227 }

```

(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

```

6228 \newrobustcmd\miko@slidelabel{
6229     \vbox to \slidelogoheight{
6230         \vss\hbox to \slidewidth
6231         {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6232     }
6233 }

```

(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.

```

6234 \def\Gin@mhrepos{}
6235 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6236 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6237 \newrobustcmd\frameimage[2][]{
6238     \stepcounter{slide}
6239     \bool_if:NT \c__notesslides_frameimages_bool {
6240         \def\Gin@ewidth{}\setkeys{Gin}{#1}
6241         \bool_if:NF \c__notesslides_notes_bool { \vfill }
6242         \begin{center}
6243             \bool_if:NTF \c__notesslides_fiboxed_bool {
6244                 \fbox{
6245                     \ifx\Gin@ewidth\@empty
6246                     \ifx\Gin@mhrepos\@empty
6247                         \mhgraphics[width=\slidewidth,#1]{#2}
6248                     \else

```

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

```

6249         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6250     \fi
6251     \else% Gin@ewidth empty
6252         \ifx\Gin@mhrepos\@empty
6253             \mhgraphics[#1]{#2}
6254         \else
6255             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6256         \fi
6257     \fi% Gin@ewidth empty
6258 }
6259 }{
6260     \ifx\Gin@ewidth\@empty
6261         \ifx\Gin@mhrepos\@empty
6262             \mhgraphics[width=\slidewidth,#1]{#2}
6263         \else
6264             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6265         \fi
6266         \ifx\Gin@mhrepos\@empty
6267             \mhgraphics[#1]{#2}
6268         \else
6269             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6270         \fi
6271     \fi% Gin@ewidth empty
6272 }
6273 \end{center}
6274 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6275 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6276 }
6277 } % 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.

```

6278 \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

```

6279 \AddToHook{begindocument}{
6280     \definecolor{green}{rgb}{0,.5,0}
6281     \definecolor{purple}{cmyk}{.3,1,0,.17}
6282 }

```

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`.

```

6283 % \def\STpresent#1{\textcolor{blue}{#1}}
6284 \def\defemph#1{\textcolor{magenta}{#1}}
6285 \def\symrefemph#1{\textcolor{cyan}{#1}}
6286 \def\compemph#1{\textcolor{blue}{#1}}
6287 \def\titleemph#1{\textcolor{blue}{#1}}
6288 \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.

```

6289 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6290 \def\smalltextwarning{
6291   \pgfuseimage{miko@small@dbend}
6292   \xspace
6293 }
6294 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6295 \newrobustcmd\textwarning{
6296   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6297   \xspace
6298 }
6299 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6300 \newrobustcmd\bigtextwarning{
6301   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6302   \xspace
6303 }
(End definition for \textwarning. This function is documented on page ??.)
6304 \newrobustcmd\putgraphicsat[3]{
6305   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6306 }
6307 \newrobustcmd\putat[2]{
6308   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6309 }

```

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`.

```

6310 \bool_if:NT \c__notesslides_sectocframes_bool {
6311   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6312     \newcounter{chapter}\counterwithin*{section}{chapter}
6313   }{
6314     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6315       \newcounter{chapter}\counterwithin*{section}{chapter}
6316     }
6317   }
6318 }

```

`\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
6319 \def\part@prefix{}
6320 \@ifpackageloaded{document-structure}{}{
6321   \str_case:VnF \__notesslidesstopsect {
6322     {part}{
6323       \int_set:Nn \l_document_structure_section_level_int {0}
6324       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6325     \def\part@prefix{\arabic{chapter}.}
6326   }
6327   {chapter}{
6328     \int_set:Nn \l_document_structure_section_level_int {1}
6329     \def\thesection{\arabic{chapter}.\arabic{section}}
6330     \def\part@prefix{\arabic{chapter}.}
6331   }
6332 }{
6333   \int_set:Nn \l_document_structure_section_level_int {2}
6334   \def\part@prefix{}
6335 }
6336 }
6337
6338 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `omgroup` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

sfragment

```

6339 \renewenvironment{sfragment}[2][]{
6340   \_document_structure_omgroup_args:n { #1 }
6341   \int_incr:N \l_document_structure_section_level_int
6342   \bool_if:NT \c__notesslides_sectocframes_bool {
6343     \stepcounter{slide}
6344     \begin{frame}[noframenumbering]
6345     \vfill\Large\centering
6346     \red{
6347       \ifcase\l_document_structure_section_level_int\or
6348         \stepcounter{part}
6349         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6350         \def\currentsectionlevel{\omdoc@part@kw}
6351       \or
6352         \stepcounter{chapter}
6353         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6354         \def\currentsectionlevel{\omdoc@chapter@kw}
6355       \or
6356         \stepcounter{section}
6357         \def\_notesslideslabel{\part@prefix\arabic{section}}
6358         \def\currentsectionlevel{\omdoc@section@kw}
6359       \or
6360         \stepcounter{subsection}
6361         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6362         \def\currentsectionlevel{\omdoc@subsection@kw}
6363       \or
6364         \stepcounter{subsubsection}
6365         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6366         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6367       \or
6368         \stepcounter{paragraph}
6369         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6370         \def\currentsectionlevel{\omdoc@paragraph@kw}
6371       \else
6372         \def\_notesslideslabel{}

```

```

6373         \def\currentsectionlevel{\omdoc@paragraph@kw}
6374         \fi% end ifcase
6375         \_notesslideslabel%\sref@label@id\_notesslideslabel
6376         \quad #2%
6377     }%
6378     \vfill%
6379     \end{frame}%
6380 }
6381 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6382     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6383 }
6384 }{}
6385 }

```

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

```

6386 \def\inserttheorembodyfont{\normalfont}
6387 %\bool_if:NF \c__notesslides_notes_bool {
6388 % \defbeamertemplate{theorem begin}{miko}
6389 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6390 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6391 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6392 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6393 % \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.

```

6394 % \expandafter\def\csname Parent2\endcsname{}
6395 %}
6396
6397 \AddToHook{begindocument}{% this does not work for some reason
6398     \setbeamertemplate{theorems}[ams style]
6399 }
6400 \bool_if:NT \c__notesslides_notes_bool {
6401     \renewenvironment{columns}[1][{}]{%
6402         \par\noindent%
6403         \begin{minipage}%
6404             \slidewidth\centering\leavevmode%
6405     }{}%
6406     \end{minipage}\par\noindent%
6407 }%
6408 \newsavebox\columnbox%
6409 \renewenvironment<>{column}[2][{}]{%
6410     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6411 }{}%
6412     \end{minipage}\end{lrbox}\usebox\columnbox%
6413 }%
6414 }
6415 \bool_if:NTF \c__notesslides_noproblems_bool {
6416     \newenvironment{problems}{}{}
6417 }{
6418     \excludecomment{problems}
6419 }

```

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.

```

6420 \gdef\printexcursions{}
6421 \newcommand\excursionref[2]{% label, text
6422   \bool_if:NT \c__notesslides_notes_bool {
6423     \begin{sparagraph}[title=Excursion]
6424       #2 \sref[fallback=the appendix]{#1}.
6425     \end{sparagraph}
6426   }
6427 }
6428 \newcommand\activate@excursion[2][]{
6429   \gappto\printexcursions{\inputref{#1}{#2}}
6430 }
6431 \newcommand\excursion[4][]{% repos, label, path, text
6432   \bool_if:NT \c__notesslides_notes_bool {
6433     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6434   }
6435 }

```

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

`\excursiongroup`

```

6436 \keys_define:nn{notesslides / excursiongroup }{
6437   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6438   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6439   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6440 }
6441 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6442   \tl_clear:N \l__notesslides_excursion_intro_tl
6443   \str_clear:N \l__notesslides_excursion_id_str
6444   \str_clear:N \l__notesslides_excursion_mhrepos_str
6445   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6446 }
6447 \newcommand\excursiongroup[1][]{
6448   \__notesslides_excursion_args:n{ #1 }
6449   \ifdefempty\printexcursions{}% only if there are excursions
6450   {\begin{note}
6451     \begin{sfragment}[#1]{Excursions}%
6452     \ifdefempty\l__notesslides_excursion_intro_tl{
6453       \inputref[\l__notesslides_excursion_mhrepos_str]{
6454         \l__notesslides_excursion_intro_tl
6455       }
6456     }
6457     \printexcursions%
6458     \end{sfragment}
6459     \end{note}}
6460 }
6461 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6462 \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.

```
6463 <*package>
6464 <@@=problems>
6465 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6466 \RequirePackage{l3keys2e,stex}
6467
6468 \keys_define:nn { problem / pkg }{
6469   notes      .default:n    = { true },
6470   notes      .bool_set:N   = \c__problems_notes_bool,
6471   gnotes     .default:n    = { true },
6472   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6473   hints      .default:n    = { true },
6474   hints      .bool_set:N   = \c__problems_hints_bool,
6475   solutions  .default:n    = { true },
6476   solutions  .bool_set:N   = \c__problems_solutions_bool,
6477   pts        .default:n    = { true },
6478   pts        .bool_set:N   = \c__problems_pts_bool,
6479   min        .default:n    = { true },
6480   min        .bool_set:N   = \c__problems_min_bool,
6481   boxed      .default:n    = { true },
6482   boxed      .bool_set:N   = \c__problems_boxed_bool,
6483   unknown    .code:n       = {}
6484 }
6485 \newif\ifsolutions
6486
6487 \ProcessKeysOptions{ problem / pkg }
6488 \bool_if:NTF \c__problems_solutions_bool {
6489   \solutionstrue
6490 }{
6491   \solutionsfalse
6492 }
```

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

```
6493 \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.

```
6494 \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.

```
6495 \def\prob@problem@kw{Problem}
6496 \def\prob@solution@kw{Solution}
6497 \def\prob@hint@kw{Hint}
6498 \def\prob@note@kw{Note}
6499 \def\prob@gnote@kw{Grading}
6500 \def\prob@pt@kw{pt}
6501 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6502 \AddToHook{begindocument}{
6503   \ltx@ifpackageloaded{babel}{
6504     \makeatletter
6505     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6506     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6507       \input{problem-ngerman.ldf}
6508     }
6509     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6510       \input{problem-finnish.ldf}
6511     }
6512     \clist_if_in:NnT \l_tmpa_clist {french}{
6513       \input{problem-french.ldf}
6514     }
6515     \clist_if_in:NnT \l_tmpa_clist {russian}{
6516       \input{problem-russian.ldf}
6517     }
6518     \makeatother
6519   }{ }
6520 }
```

40.2 Problems and Solutions

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

```
6521 \keys_define:nn{ problem / problem }{
6522   id      .str_set:x:N = \l__problems_prob_id_str,
6523   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6524   min     .tl_set:N    = \l__problems_prob_min_tl,
6525   title   .tl_set:N    = \l__problems_prob_title_tl,
6526   type    .tl_set:N    = \l__problems_prob_type_tl,
6527   refnum  .int_set:N    = \l__problems_prob_refnum_int
6528 }
6529 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6530 \str_clear:N \l__problems_prob_id_str
6531 \tl_clear:N \l__problems_prob_pts_tl
6532 \tl_clear:N \l__problems_prob_min_tl
6533 \tl_clear:N \l__problems_prob_title_tl
6534 \tl_clear:N \l__problems_prob_type_tl
6535 \int_zero_new:N \l__problems_prob_refnum_int
6536 \keys_set:nn { problem / problem }{ #1 }
6537 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6538   \let\l__problems_prob_refnum_int\undefined
6539 }
6540 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6541 \newcounter{problem}
6542 \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.

```

6543 \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

```

6544 \newcommand\prob@number{
6545   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6546     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6547   }{
6548     \int_if_exist:NTF \l__problems_prob_refnum_int {
6549       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6550     }{
6551       \prob@label\theproblem
6552     }
6553   }
6554 }

```

(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.

```

6555 \newcommand\prob@title[3]{%
6556   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6557     #2 \l__problems_inclprob_title_tl #3
6558   }{
6559     \tl_if_exist:NTF \l__problems_prob_title_tl {
6560       #2 \l__problems_prob_title_tl #3
6561     }{
6562       #1
6563     }
6564   }
6565 }

```

(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.

```

6566 \def\prob@heading{
6567   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6568   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6569 }

```

(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

```

6570 \newenvironment{sproblem}[1][{}]{
6571   \__problems_prob_args:n{#1}%\sref@target%
6572   \@in@omtexttrue% we are in a statement (for inline definitions)
6573   \stepcounter{problem}\record@problem
6574   \def\current@section@level{\prob@problem@kw}
6575   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6576     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6577   }{
6578     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6579   }
6580   \str_if_exist:NTF \l__problems_inclprob_id_str {
6581     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6582   }{
6583     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6584   }
6585
6586
6587   \clist_set:No \l_tmpa_clist \sproblemtype
6588   \tl_clear:N \l_tmpa_tl
6589   \clist_map_inline:Nn \l_tmpa_clist {
6590     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6591       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6592     }
6593   }
6594   \tl_if_empty:NTF \l_tmpa_tl {
6595     \__problems_sproblem_start:
6596   }{
6597     \l_tmpa_tl
6598   }
6599   \stex_ref_new_doc_target:n \sproblemid
6600 }{
6601   \clist_set:No \l_tmpa_clist \sproblemtype
6602   \tl_clear:N \l_tmpa_tl
6603   \clist_map_inline:Nn \l_tmpa_clist {
6604     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6605       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6606     }

```



```

6607 }
6608 \tl_if_empty:NTF \l_tmpa_tl {
6609   \__problems_sproblem_end:
6610 }{
6611   \l_tmpa_tl
6612 }
6613
6614
6615 \smallskip
6616 }
6617
6618
6619 \cs_new_protected:Nn \__problems_sproblem_start: {
6620   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6621 }
6622 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6623
6624 \newcommand\stexpatchproblem[3][] {
6625   \str_set:Nx \l_tmpa_str{ #1 }
6626   \str_if_empty:NTF \l_tmpa_str {
6627     \tl_set:Nn \__problems_sproblem_start: { #2 }
6628     \tl_set:Nn \__problems_sproblem_end: { #3 }
6629   }{
6630     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6631     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6632   }
6633 }
6634
6635
6636 \bool_if:NT \c__problems_boxed_bool {
6637   \surroundwithmdframed{problem}
6638 }

```

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

```

6639 \def\record@problem{
6640   \protected@write\@auxout{}
6641   {
6642     \string\@problem{\prob@number}
6643     {
6644       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6645         \l__problems_inclprob_pts_tl
6646       }{
6647         \l__problems_prob_pts_tl
6648       }
6649     }%
6650     {
6651       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6652         \l__problems_inclprob_min_tl
6653       }{
6654         \l__problems_prob_min_tl
6655       }
6656     }
6657   }
6658 }

```

(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).

```
6659 \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.

```
6660 \keys_define:nn { problem / solution }{
6661   id                .str_set_x:N = \l__problems_solution_id_str ,
6662   for               .tl_set:N   = \l__problems_solution_for_tl ,
6663   height            .dim_set:N  = \l__problems_solution_height_dim ,
6664   creators          .clist_set:N = \l__problems_solution_creators_clist ,
6665   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
6666   srccite           .tl_set:N   = \l__problems_solution_srccite_tl
6667 }
6668 \cs_new_protected:Nn \__problems_solution_args:n {
6669   \str_clear:N \l__problems_solution_id_str
6670   \tl_clear:N \l__problems_solution_for_tl
6671   \tl_clear:N \l__problems_solution_srccite_tl
6672   \clist_clear:N \l__problems_solution_creators_clist
6673   \clist_clear:N \l__problems_solution_contributors_clist
6674   \dim_zero:N \l__problems_solution_height_dim
6675   \keys_set:nn { problem / solution }{ #1 }
6676 }
```

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

```
6677 \newcommand\@startsolution[1][{}]{
6678   \__problems_solution_args:n { #1 }
6679   \@in@omtexttrue% we are in a statement.
6680   \bool_if:NF \c__problems_boxed_bool { \hrule }
6681   \smallskip\noindent
6682   {\textbf\prob@solution@kw : \enspace}
6683   \begin{small}
6684   \def\current@section@level{\prob@solution@kw}
6685   \ignorespacesandpars
6686 }
```

\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.

```
6687 \newcommand\startsolutions{
6688   \specialcomment{solution}{\@startsolution}{
6689     \bool_if:NF \c__problems_boxed_bool {
6690       \hrule\medskip
6691     }
6692     \end{small}%
6693   }
6694   \bool_if:NT \c__problems_boxed_bool {
6695     \surroundwithmdframed{solution}
6696   }
6697 }
```

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

\stopsolutions

```
6698 \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.

```
6699 \ifsolutions
6700 \startsolutions
6701 \else
6702 \stopsolutions
6703 \fi
```

exnote

```
6704 \bool_if:NTF \c__problems_notes_bool {
6705 \newenvironment{exnote}[1][ ]{
6706 \par\smallskip\hrule\smallskip
6707 \noindent\textbf{\prob@note@kw : }\small
6708 }{
6709 \smallskip\hrule
6710 }
6711 }{
6712 \excludecomment{exnote}
6713 }
```

hint

```
6714 \bool_if:NTF \c__problems_notes_bool {
6715 \newenvironment{hint}[1][ ]{
6716 \par\smallskip\hrule\smallskip
6717 \noindent\textbf{\prob@hint@kw :~ }\small
6718 }{
6719 \smallskip\hrule
6720 }
6721 \newenvironment{exhint}[1][ ]{
6722 \par\smallskip\hrule\smallskip
6723 \noindent\textbf{\prob@hint@kw :~ }\small
6724 }{
6725 \smallskip\hrule
6726 }
6727 }{
6728 \excludecomment{hint}
6729 \excludecomment{exhint}
6730 }
```

gnote

```
6731 \bool_if:NTF \c__problems_notes_bool {
6732 \newenvironment{gnote}[1][ ]{
6733 \par\smallskip\hrule\smallskip
6734 \noindent\textbf{\prob@gnote@kw : }\small
6735 }{
6736 \smallskip\hrule
6737 }
6738 }{
6739 \excludecomment{gnote}
6740 }
```

40.3 Multiple Choice Blocks

EdN:22

mcb 22

```

6741 \newenvironment{mcb}{
6742   \begin{enumerate}
6743 }{
6744   \end{enumerate}
6745 }
```

we define the keys for the mcc macro

```

6746 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6747   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6748     \bool_set_true:N #1
6749   }{
6750     \bool_set_false:N #1
6751   }
6752 }
6753 \keys_define:nn { problem / mcc }{
6754   id          .str_set:x:N = \l__problems_mcc_id_str ,
6755   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6756   T           .default:n    = { true } ,
6757   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6758   F           .default:n    = { true } ,
6759   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6760   Ttext       .code:n       = {
6761     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6762   } ,
6763   Ftext       .code:n       = {
6764     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6765   }
6766 }
6767 \cs_new_protected:Nn \l__problems_mcc_args:n {
6768   \str_clear:N \l__problems_mcc_id_str
6769   \tl_clear:N \l__problems_mcc_feedback_tl
6770   \bool_set_true:N \l__problems_mcc_t_bool
6771   \bool_set_true:N \l__problems_mcc_f_bool
6772   \bool_set_true:N \l__problems_mcc_Ttext_bool
6773   \bool_set_false:N \l__problems_mcc_Ftext_bool
6774   \keys_set:nn { problem / mcc }{ #1 }
6775 }
```

\mcc

```

6776 \newcommand\mcc[2][] {
6777   \l__problems_mcc_args:n{ #1 }
6778   \item #2
6779   \ifsolutions
6780     \\\
6781     \bool_if:NT \l__problems_mcc_t_bool {
6782       % TODO!
6783       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6784     }
6785     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

6786      % TODO!
6787      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6788    }
6789    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6790      !
6791    }{
6792      \l__problems_mcc_feedback_tl
6793    }
6794    \fi
6795  } %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.

```

6796
6797 \keys_define:nn{ problem / inclproblem }{
6798   id      .str_set:N = \l__problems_inclprob_id_str,
6799   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6800   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6801   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6802   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6803   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6804   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6805 }
6806 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6807   \str_clear:N \l__problems_prob_id_str
6808   \tl_clear:N \l__problems_inclprob_pts_tl
6809   \tl_clear:N \l__problems_inclprob_min_tl
6810   \tl_clear:N \l__problems_inclprob_title_tl
6811   \tl_clear:N \l__problems_inclprob_type_tl
6812   \int_zero_new:N \l__problems_inclprob_refnum_int
6813   \str_clear:N \l__problems_inclprob_mhrepos_str
6814   \keys_set:nn { problem / inclproblem }{ #1 }
6815   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6816     \let\l__problems_inclprob_pts_tl\undefined
6817   }
6818   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6819     \let\l__problems_inclprob_min_tl\undefined
6820   }
6821   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6822     \let\l__problems_inclprob_title_tl\undefined
6823   }
6824   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6825     \let\l__problems_inclprob_type_tl\undefined
6826   }
6827   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6828     \let\l__problems_inclprob_refnum_int\undefined
6829   }
6830 }

```

```

6831
6832 \cs_new_protected:Nn \__problems_inclprob_clear: {
6833   \let\l__problems_inclprob_id_str\undefined
6834   \let\l__problems_inclprob_pts_tl\undefined
6835   \let\l__problems_inclprob_min_tl\undefined
6836   \let\l__problems_inclprob_title_tl\undefined
6837   \let\l__problems_inclprob_type_tl\undefined
6838   \let\l__problems_inclprob_refnum_int\undefined
6839   \let\l__problems_inclprob_mhrepos_str\undefined
6840 }
6841 \__problems_inclprob_clear:
6842
6843 \newcommand\includeproblem[2][ ]{
6844   \__problems_inclprob_args:n{ #1 }
6845   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6846     \input{#2}
6847   }{
6848     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6849       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6850     }
6851   }
6852   \__problems_inclprob_clear:
6853 }

```

(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.

```

6854 \AddToHook{enddocument}{
6855   \bool_if:NT \c__problems_pts_bool {
6856     \message{Total:~\arabic{pts}~points}
6857   }
6858   \bool_if:NT \c__problems_min_bool {
6859     \message{Total:~\arabic{min}~minutes}
6860   }
6861 }

```

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

```

6862 \def\pts#1{
6863   \bool_if:NT \c__problems_pts_bool {
6864     \marginpar{#1~\prob@pt@kw}
6865   }
6866 }
6867 \def\min#1{
6868   \bool_if:NT \c__problems_min_bool {
6869     \marginpar{#1~\prob@min@kw}
6870   }
6871 }

```

`\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.

```

6872 \newcounter{pts}
6873 \def\show@pts{
6874   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6875     \bool_if:NT \c__problems_pts_bool {
6876       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6877       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6878     }
6879   }{
6880     \tl_if_exist:NT \l__problems_prob_pts_tl {
6881       \bool_if:NT \c__problems_pts_bool {
6882         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6883         \addtocounter{pts}{\l__problems_prob_pts_tl}
6884       }
6885     }
6886   }
6887 }

```

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

`\show@min`

```

6888 \newcounter{min}
6889 \def\show@min{
6890   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6891     \bool_if:NT \c__problems_min_bool {
6892       \marginpar{\l__problems_inclprob_min_tl\ min}
6893       \addtocounter{min}{\l__problems_inclprob_min_tl}
6894     }
6895   }{
6896     \tl_if_exist:NT \l__problems_prob_min_tl {
6897       \bool_if:NT \c__problems_min_bool {
6898         \marginpar{\l__problems_prob_min_tl\ min}
6899         \addtocounter{min}{\l__problems_prob_min_tl}
6900       }
6901     }
6902   }
6903 }
6904 \</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.

```
6905 \@@=hwexam>
6906 \*cls>
6907 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
6908 \RequirePackage{l3keys2e}
6909 \DeclareOption*{
6910   \PassOptionsToClass{\CurrentOption}{document-structure}
6911   \PassOptionsToPackage{\CurrentOption}{stex}
6912   \PassOptionsToPackage{\CurrentOption}{hwexam}
6913   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6914 }
6915 \ProcessOptions
```

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

```
6916 \LoadClass{document-structure}
6917 \RequirePackage{stex}
6918 \RequirePackage{hwexam}
6919 \RequirePackage{tikzinput}
6920 \RequirePackage{graphicx}
6921 \RequirePackage{a4wide}
6922 \RequirePackage{amssymb}
6923 \RequirePackage{amstext}
6924 \RequirePackage{amsmath}
```

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


```

6925 \newcommand\assig@default@type{\hwexam@assignment@kw}
6926 \def\document@hwexamtype{\assig@default@type}
6927 <@@=document_structure>
6928 \keys_define:nn { document-structure / document }{
6929 id .str_set_x:N = \c_document_structure_document_id_str,
6930 hwexamtype .tl_set:N = \document@hwexamtype
6931 }
6932 <@@=hwexam>
6933 </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.

```
6934 \*package>
6935 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
6936 \RequirePackage{13keys2e}
6937
6938 \newif\iftest\testfalse
6939 \DeclareOption{test}{\testtrue}
6940 \newif\ifmultiple\multiplefalse
6941 \DeclareOption{multiple}{\multipletrue}
6942 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6943 \ProcessOptions
```

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

```
6944 \RequirePackage{keyval}[1997/11/10]
6945 \RequirePackage{problem}
```

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

```
6946 \newcommand\hwexam@assignment@kw{Assignment}
6947 \newcommand\hwexam@given@kw{Given}
6948 \newcommand\hwexam@due@kw{Due}
6949 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6950 blank~for~extra~space}
6951 \def\hwexam@minutes@kw{minutes}
6952 \newcommand\correction@probs@kw{prob.}
6953 \newcommand\correction@pts@kw{total}
6954 \newcommand\correction@reached@kw{reached}
6955 \newcommand\correction@sum@kw{Sum}
6956 \newcommand\correction@grade@kw{grade}
6957 \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

```

6958 \AddToHook{begindocument}{
6959 \ltx@ifpackageloaded{babel}{
6960 \makeatletter
6961 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6962 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6963 \input{hwexam-ngerman.ldf}
6964 }
6965 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6966 \input{hwexam-finnish.ldf}
6967 }
6968 \clist_if_in:NnT \l_tmpa_clist {french}{
6969 \input{hwexam-french.ldf}
6970 }
6971 \clist_if_in:NnT \l_tmpa_clist {russian}{
6972 \input{hwexam-russian.ldf}
6973 }
6974 \makeatother
6975 }{}
6976 }
6977

```

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.

```

6978 \newcounter{assignment}
6979 \numberproblemsin{assignment}
6980 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6981 \keys_define:nn { hwexam / assignment } {
6982 id .str_set:N = \l__hwexam_assign_id_str,
6983 number .int_set:N = \l__hwexam_assign_number_int,
6984 title .tl_set:N = \l__hwexam_assign_title_tl,
6985 type .tl_set:N = \l__hwexam_assign_type_tl,
6986 given .tl_set:N = \l__hwexam_assign_given_tl,
6987 due .tl_set:N = \l__hwexam_assign_due_tl,
6988 loadmodules .code:n = {
6989 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6990 }
6991 }
6992 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6993 \str_clear:N \l__hwexam_assign_id_str
6994 \int_set:Nn \l__hwexam_assign_number_int {-1}
6995 \tl_clear:N \l__hwexam_assign_title_tl
6996 \tl_clear:N \l__hwexam_assign_type_tl
6997 \tl_clear:N \l__hwexam_assign_given_tl
6998 \tl_clear:N \l__hwexam_assign_due_tl
6999 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7000 \keys_set:nn { hwexam / assignment }{ #1 }
7001 }

```

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.

```

7002 \newcommand\given@due[2]{
7003 \bool_lazy_all:nF {
7004 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7005 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7006 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7007 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7008 }{ #1 }
7009
7010 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7011 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7012 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7013 }
7014 }{
7015 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7016 }
7017
7018 \bool_lazy_or:nnF {
7019 \bool_lazy_and_p:nn {
7020 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7021 }{
7022 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7023 }
7024 }{
7025 \bool_lazy_and_p:nn {
7026 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7027 }{
7028 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7029 }
7030 }{ ,~ }
7031
7032 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7033 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7034 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7035 }
7036 }{
7037 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7038 }
7039
7040 \bool_lazy_all:nF {
7041 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7042 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7043 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7044 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7045 }{ #2 }
7046 }

```

`\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.

```

7047 \newcommand\assignment@title[3]{
7048 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
7049 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7050 #1
7051 }{
7052 #2\l__hwexam_assign_title_tl#3
7053 }
7054 }{
7055 #2\l__hwexam_inclassassign_title_tl#3
7056 }
7057 }

```

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

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

```

7058 \newcommand\assignment@number{
7059 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
7060 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7061 \arabic{assignment}
7062 } {
7063 \int_use:N \l__hwexam_assign_number_int
7064 }
7065 }{
7066 \int_use:N \l__hwexam_inclassassign_number_int
7067 }
7068 }

```

(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.

```

7069 \newenvironment{assignment}[1][ ]{
7070 \__hwexam_assignment_args:n { #1 }
7071 %\sref@target
7072 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7073 \global\stepcounter{assignment}
7074 }{
7075 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7076 }
7077 \setcounter{problem}{0}
7078 \def\current@section@level{\document@hwexamtype}
7079 %\sref@label@id{\document@hwexamtype \thesection}
7080 \begin{@assignment}
7081 }{
7082 \end{@assignment}
7083 }

```

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

```

7084 \def\ass@title{
7085 \protect\document@hwexamtype~\arabic{assignment}
7086 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7087 }
7088 \ifmultiple
7089 \newenvironment{@assignment}{
7090 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7091 \begin{sfragment}[loadmodules]{\ass@title}
7092 }{
7093 \begin{sfragment}{\ass@title}
7094 }
7095 }{
7096 \end{sfragment}
7097 }

```

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

```

7098 \else
7099 \newenvironment{@assignment}{
7100 \begin{center}\bf
7101 \Large@title\strut\
7102 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7103 \large\given@due{--;\}{}{;\}{}
7104 \end{center}
7105 }{}
7106 \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.

```

7107 \keys_define:nn { hwexam / inclassignment } {
7108 %id .str_set_x:N = \l__hwexam_assign_id_str,
7109 number .int_set:N = \l__hwexam_inclassign_number_int,
7110 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7111 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7112 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7113 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7114 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7115 }
7116 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7117 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7118 \tl_clear:N \l__hwexam_inclassign_title_tl
7119 \tl_clear:N \l__hwexam_inclassign_type_tl
7120 \tl_clear:N \l__hwexam_inclassign_given_tl
7121 \tl_clear:N \l__hwexam_inclassign_due_tl
7122 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7123 \keys_set:nn { hwexam / inclassignment }{ #1 }
7124 }
7125 \__hwexam_inclassignment_args:n {}
7126
7127 \newcommand\inputassignment[2][{}]{

```

```

7128 \_hwexam_inclassnment_args:n { #1 }
7129 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7130 \input{#2}
7131 }{
7132 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7133 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7134 }
7135 }
7136 \_hwexam_inclassnment_args:n {}
7137 }
7138 \newcommand\includeassignment[2][ ]{
7139 \newpage
7140 \inputassignment[#1]{#2}
7141 }

```

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

42.4 Typesetting Exams

\quizheading

```

7142 \ExplSyntaxOff
7143 \newcommand\quizheading[1]{%
7144 \def\@tas{#1}%
7145 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7146 \ifx\@tas\@empty\else%
7147 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7148 \fi%
7149 }
7150 \ExplSyntaxOn

```

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

\testheading

```

7151
7152 \def\hwexamheader{\input{hwexam-default.header}}
7153
7154 \def\hwexamminutes{
7155 \tl_if_empty:NTF \testheading@duration {
7156 {\testheading@min}~\hwexam@minutes@kw
7157 }{
7158 \testheading@duration
7159 }
7160 }
7161
7162 \keys_define:nn { hwexam / testheading } {
7163 min .tl_set:N = \testheading@min,
7164 duration .tl_set:N = \testheading@duration,
7165 reqpts .tl_set:N = \testheading@reqpts,
7166 tools .tl_set:N = \testheading@tools
7167 }
7168 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7169 \tl_clear:N \testheading@min
7170 \tl_clear:N \testheading@duration

```

```

7171 \tl_clear:N \testheading@reqpts
7172 \tl_clear:N \testheading@tools
7173 \keys_set:nn { hwexam / testheading }{ #1 }
7174 }
7175 \newenvironment{testheading}[1][]{
7176   \_hwexam_testheading_args:n{ #1 }
7177   \newcount\check@time\check@time=\testheading@min
7178   \advance\check@time by -\theassignment@totalmin
7179   \newif\if@bonuspoints
7180   \tl_if_empty:NTF \testheading@reqpts {
7181     \@bonuspointsfalse
7182   }{
7183     \newcount\bonus@pts
7184     \bonus@pts=\theassignment@totalpts
7185     \advance\bonus@pts by -\testheading@reqpts
7186     \edef\bonus@pts{\the\bonus@pts}
7187     \@bonuspointstrue
7188   }
7189   \edef\check@time{\the\check@time}
7190
7191   \makeatletter\hwexamheader\makeatother
7192 }{
7193   \newpage
7194 }

```

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

\testspace

```

7195 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7196 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7197 \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.

```

7198 <@=problems>
7199 \renewcommand\@problem[3]{
7200   \stepcounter{assignment@probs}
7201   \def\__problemspts{#2}
7202   \ifx\__problemspts\@empty\else
7203     \addtocounter{assignment@totalpts}{#2}
7204   \fi
7205   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7206   \xdef\correction@probs{\correction@probs & #1}%
7207   \xdef\correction@pts{\correction@pts & #2}
7208   \xdef\correction@reached{\correction@reached &}

```



```

7209 }
7210 <@@=hwexam>

```

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

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

```

7211 \newcounter{assignment@probs}
7212 \newcounter{assignment@totalpts}
7213 \newcounter{assignment@totalmin}
7214 \def\correction@probs{\correction@probs@kw}
7215 \def\correction@pts{\correction@pts@kw}
7216 \def\correction@reached{\correction@reached@kw}
7217 \stepcounter{assignment@probs}
7218 \newcommand\correction@table{
7219 \resizebox{\textwidth}{!}{%
7220 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7221 &\multicolumn{\theassignment@probs}{c|}|%|
7222 {\footnotesize\correction@forgrading@kw} &\\ \hline
7223 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7224 \correction@pts & \theassignment@totalpts & \\ \hline
7225 \correction@reached & & \[.7cm]\hline
7226 \end{tabular}}
7227 </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}

```