

The sTeX3 Package *

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

2022-02-23

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 2022-02-23)

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 \LaTeX	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	<code>sTeX</code>-Basics Implementation	69
25.1	The <code>sTeXDocument</code> 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	<code>sTeX</code>-MathHub Implementation	76
26.1	Generic Path Handling	76
26.2	PWD and <code>kpsewhich</code>	78
26.3	File Hooks and Tracking	79
26.4	MathHub Repositories	80
26.5	Using Content in Archives	84
27	<code>sTeX</code>-References Implementation	89
27.1	Document URIs and URLs	89
27.2	Setting Reference Targets	91
27.3	Using References	93
28	<code>sTeX</code>-Modules Implementation	96
28.1	The <code>smodule</code> environment	100
28.2	Invoking modules	105
29	<code>sTeX</code>-Module Inheritance Implementation	107
29.1	SMS Mode	107
29.2	Inheritance	110
30	<code>sTeX</code>-Symbols Implementation	115
30.1	Symbol Declarations	115
30.2	Notations	122
30.3	Variables	131
31	<code>sTeX</code>-Terms Implementation	136
31.1	Symbol Invocations	136
31.2	Terms	141
31.3	Notation Components	148
31.4	Variables	150
32	<code>sTeX</code>-Structural Features Implementation	152
32.1	Imports with modification	152
32.2	The feature environment	159
32.3	Features	160
33	<code>sTeX</code>-Statements Implementation	166
33.1	Definitions	166
33.2	Assertions	171
33.3	Examples	174
33.4	Logical Paragraphs	177

34 The Implementation	181
34.1 Package Options	181
34.2 Proofs	181
34.3 Justifications	192
35 \LaTeX-Others Implementation	193
36 \LaTeX-Metatheory Implementation	194
37 Tikzinput Implementation	197
38 document-structure.sty Implementation	199
38.1 The document-structure Class	199
38.2 Class Options	199
38.3 Beefing up the <code>document</code> environment	200
38.4 Implementation: document-structure Package	200
38.5 Package Options	200
38.6 Document Structure	202
38.7 Front and Backmatter	205
38.8 Global Variables	207
39 NotesSlides – Implementation	208
39.1 Class and Package Options	208
39.2 Notes and Slides	210
39.3 Header and Footer Lines	214
39.4 Frame Images	215
39.5 Colors and Highlighting	216
39.6 Sectioning	217
39.7 Excursions	219
40 The Implementation	221
40.1 Package Options	221
40.2 Problems and Solutions	222
40.3 Multiple Choice Blocks	228
40.4 Including Problems	229
40.5 Reporting Metadata	230
41 Implementation: The hwexam Class	232
41.1 Class Options	232
42 Implementation: The hwexam Package	234
42.1 Package Options	234
42.2 Assignments	235
42.3 Including Assignments	238
42.4 Typesetting Exams	239
42.5 Leftovers	241

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with `pdflatex` should yield the output

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

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

\usemodule

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

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

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using sTeX

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

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

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

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

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

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

TODO

Chapter 4

TeX Archives

4.1 The Local MathHub-Directory

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

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

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

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

```
\begin{smodule}{assoctest}
\symdef{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: $a:w_1; b:w_2; c:[w_1; x+[w_1; y+z; w_2]; w_2]$

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]{\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)

Chapter 6

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

Chapter 7

Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

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

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 5

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

ab

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

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

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

Example 6

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

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

.

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

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

Example 7

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

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

.

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

Example 8

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

Multiplying again by b yields...

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

Example 9

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

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

.

⁴EdNOTE: TODO

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

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

Example 10

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

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

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

Example 11

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

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

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

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

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

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

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

Other Argument Types

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

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

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

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

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

Example 12

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

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

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

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

Example 13

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

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

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

5 6

⁵EDNOTE: what about e.g. `\int _x \int _y \int _z f dx dy dz`?

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

Precedences

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

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

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

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

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

Consequently, if some operator A should bind stronger than some operator B , then A 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 <i>current</i> MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the <code>MANIFEST.MF</code> -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 <code>__stex_mathhub_do_manifest:n</code> , so works whether this repository's <code>MANIFEST.MF</code> -file has already been read or not.
---	---

<code>\stex_require_repository:n</code>	Calls <code>__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
---	--

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

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

10.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<archive-ID>}{<filename>}</code>
	Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\mhinput</code> <hr/>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-</code> mode, inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code> <hr/>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code> <hr/>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <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>assocs</code> (integer string; number of associative arguments),
<hr/> <hr/> <code>\l_stex_all_symbols_seq</code>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_get_symbol:n</code>	Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{<URI>}{<notations⁺>}</code></p> <p>Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>.</p> <p>Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>#<variant>#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code> (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code></p> <p>Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

<hr/> <code>\infprec</code> <hr/> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<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/> <code>\withbrackets</code> <hr/>	<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/> <code>\stex_term_custom:nn</code> <hr/>	<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/> <code>\stex_highlight_term:nn</code> <hr/>	<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/> <code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <hr/>	<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/> <code>\STEXinvisible</code> <hr/>	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/> <code>\ellipses</code> <hr/>	TODO

Chapter 16

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

```
\begin{sproof}[id=simple-proof]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcases}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[type=inline] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[type=inline]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[type=inline] We compute  $1+3=2^2=4$ .\end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n>1$ }
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ .
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for  $n=k+1$  from
        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
      \end{sproofcomment}
      \begin{spfstep}
        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to  $(k+1)^2$ , which proves the assertion.
      \end{spfstep}
    \end{spfcase}
  \end{spfcases}
\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 $1 = 1^2$ □
 - 1.3. $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □
 - 1.5. $n > 1$:
 - 1.6. Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.
 - 1.7. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.
 - 1.7. We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum
 - 1.8. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.
 - 1.9. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □
 - 1.10. 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.

EdN:8

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

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

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

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

21.2.5 Global Variables

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

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

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

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

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

22.2.1 Package Options

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

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

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

22.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

22.2.3 Header and Footer Lines of the Slides

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

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

22.2.4 Frame Images

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

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

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

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

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


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

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

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 Excursions

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

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

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

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

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

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

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

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

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


22.2.8 Miscellaneous

22.3 Limitations

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

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

Chapter 23

problem.sty: An Infrastructure for formatting Problems

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

23.1 Introduction

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

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

23.2 The User Interface

23.2.1 Package Options

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

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

23.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

25.3 Messages and logging

```

41 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

25.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`
`\latexml_if:TF`

```

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

```

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

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

```

89 \tl_new:N \l__stex_annotate_arg_tl
90 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
91   \rustex_if:TF {
92     \rustex_direct_HTML:n { \c_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 }

```

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

`\stex_filestack_pop:`

```

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

```

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

Hooks for the current file:

```

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

```

26.4 MathHub Repositories

420 `<@=stex_mathhub>`

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

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

```

421 \str_if_empty:NTF\mathhub{
422   \sys_if_platform_windows:TF{
423     \begingroup\escapechar=-1\catcode'\=12
424     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
425     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
426     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
427   }{
428     \stex_kpsewhich:n{-var-value-MATHHUB}
429   }
430   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
431 }
432 \str_if_empty:NTF\c_stex_mathhub_str{
433   \msg_warning:nn{stex}{warning/nomathhub}
434 }{
435   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
436   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
437 }
438 }{
439   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
440   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
441     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
442       \c_stex_pwd_str/\mathhub
443     }
444   }
445 }

```

```

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   \tl_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.sty}
712   \IfFileExists{ \l_tmpa_str }{
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.sty}
720 \IfFileExists{ \l_tmpa_str }{
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      = \smoduletitle ,
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 \smoduletitle
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 }

```

(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}`

```

1347 \cs_new_protected:Nn \__stex_modules_begin_module: {
1348   \stex_reactivate_macro:N \STEXexport
1349   \stex_reactivate_macro:N \importmodule
1350   \stex_reactivate_macro:N \symdecl
1351   \stex_reactivate_macro:N \notation
1352   \stex_reactivate_macro:N \symdef
1353
1354   \stex_debug:nn{modules}{
1355     New~module:\\
1356     Namespace:~\l_stex_module_ns_str\\
1357     Name:~\l_stex_module_name_str\\
1358     Language:~\l_stex_module_lang_str\\
1359     Signature:~\l_stex_module_sig_str\\
1360     Metatheory:~\l_stex_module_meta_str\\
1361     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1362   }
1363
1364   \seq_put_right:Nx \l_stex_all_modules_seq {
1365     \l_stex_module_ns_str ? \l_stex_module_name_str
1366   }
1367
1368   \stex_if_smsmode:F{
1369     \begin{stex_annotate_env} {theory} {
1370       \l_stex_module_ns_str ? \l_stex_module_name_str
1371     }
1372
1373     \stex_annotate_invisible:nnn{header}{} {
1374       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1375       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1376       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1377         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1378       }
1379       \str_if_empty:NF \smoduletype {
1380         \stex_annotate:nnn{type}{\smoduletype}{}
1381       }
1382     }
1383   }
1384   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1385   % TODO: Inherit metatheory for nested modules?
1386 }
1387 \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{smodule}`

```

1388 \cs_new_protected:Nn \__stex_modules_end_module: {

```

```

1389 \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module
1390 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1391 \iffalse \begin{stex_annotate_env} \fi %%^A make syntax highlighting work again
1392 \NewDocumentEnvironment { smodule } { 0{} m } {
1393   \stex_module_setup:nn{#1}{#2}
1394   \par
1395   \stex_if_smsmode:F{
1396     \tl_clear:N \l_tmpa_tl
1397     \clist_map_inline:Nn \smoduletype {
1398       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1399         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1400       }
1401     }
1402     \tl_if_empty:NTF \l_tmpa_tl {
1403       \__stex_modules_smodule_start:
1404     }{
1405       \l_tmpa_tl
1406     }
1407   }
1408   \__stex_modules_begin_module:
1409   \str_if_empty:NF \smoduleid {
1410     \stex_ref_new_doc_target:n \smoduleid
1411   }
1412   \stex_smsmode_do:
1413 } {
1414   \__stex_modules_end_module:
1415   \stex_if_smsmode:F {
1416     \end{stex_annotate_env}
1417     \clist_set:Nn \l_tmpa_clist \smoduletype
1418     \tl_clear:N \l_tmpa_tl
1419     \clist_map_inline:Nn \l_tmpa_clist {
1420       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1421         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1422       }
1423     }
1424     \tl_if_empty:NTF \l_tmpa_tl {
1425       \__stex_modules_smodule_end:
1426     }{
1427       \l_tmpa_tl
1428     }
1429   }
1430 }

```

\stexpatchmodule

```

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

```

```

1438     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1439   }{
1440     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1441     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1442   }
1443 }

```

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

28.2 Invoking modules

```

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

```

```

1485 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1486   \stex_invoke_symbol:n{#1?#2}
1487 }

```

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

`\stex_activate_module:n`

```

1488 \bool_new:N \l_stex_in_meta_bool
1489 \bool_set_false:N \l_stex_in_meta_bool
1490 \cs_new_protected:Nn \stex_activate_module:n {
1491   \stex_debug:nn{modules}{Activating~module~#1}
1492   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1493     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1494   }
1495   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1496     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1497     \use:c{ c_stex_module_#1_code }
1498   }
1499 }

```

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

```

1500 \</package>

```


Chapter 29

STEX -Module Inheritance Implementation

```
1501 <*package>
1502
1503 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1504
```

29.1 SMS Mode

```
1505 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1506 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1507 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1508 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1509
1510 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1511   \makeatletter
1512   \makeatother
1513   \ExplSyntaxOn
1514   \ExplSyntaxOff
1515   \rustexBREAK
1516 }
1517
1518 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1519   \symdef
1520   \importmodule
1521   \notation
1522   \symdecl
1523   \STEXexport
1524   \inlineass
1525   \inlinedef
1526   \inlineex
1527   \endinput
1528   \setnotation
```

```

1529 \copynotation
1530 }
1531
1532 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1533   \tl_to_str:n {
1534     smodule,
1535     copymodule,
1536     interpretmodule,
1537     sdefinition,
1538     sexample,
1539     sassertion,
1540     sparagraph
1541   }
1542 }

```

(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`

```

1543 \bool_new:N \g__stex_smsmode_bool
1544 \bool_set_false:N \g__stex_smsmode_bool
1545 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1546   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1547 }

```

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

`__stex_smsmode_in_smsmode:nn`

```

1548 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:nn {
1549   \vbox_set:Nn \l_tmpa_box {
1550     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1551     \bool_gset_true:N \g__stex_smsmode_bool
1552     #2
1553     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1554   }
1555   \box_clear:N \l_tmpa_box
1556 }

```

(End definition for `__stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1557 \quark_new:N \q__stex_smsmode_break
1558
1559 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1560   \stex_filestack_push:n{#1}
1561   \__stex_smsmode_in_smsmode:nn{#1} {
1562     #2
1563     \everyeof{\q__stex_smsmode_break\noexpand}
1564     \expandafter\expandafter\expandafter
1565     \stex_smsmode_do:
1566     \csname @ @ input\endcsname "#1"\relax
1567   }
1568   \stex_filestack_pop:
1569 }

```

(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:

```

1570 \cs_new_protected:Npn \stex_smsmode_do: {
1571   \stex_if_smsmode:T {
1572     \__stex_smsmode_do:w
1573   }
1574 }
1575 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1576   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1577     \expandafter\if\expandafter\relax\noexpand#1
1578     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1579   } \else\expandafter\__stex_smsmode_do:w\fi
1580 }{
1581   \__stex_smsmode_do:w % #1
1582 }
1583 }
1584 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1585   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1586     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1587       #1\__stex_smsmode_do:w
1588     }{
1589       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1590         #1
1591       }{
1592         \cs_if_eq:NNTF \begin #1 {
1593           \__stex_smsmode_check_begin:n
1594         }{
1595           \cs_if_eq:NNTF \end #1 {
1596             \__stex_smsmode_check_end:n
1597           }{
1598             \__stex_smsmode_do:w
1599           }
1600         }
1601       }
1602     }
1603   }
1604 }
1605
1606 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1607   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1608     \begin{#1}
1609   }{
1610     \__stex_smsmode_do:w
1611   }
1612 }
1613 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1614   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1615     \end{#1}\__stex_smsmode_do:w
1616   }{
1617     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1618   }
1619 }

```

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

29.2 Inheritance

```

1620 <@@=stex_importmodule>

\stex_import_module_uri:nn

1621 \cs_new_protected:Nn \stex_import_module_uri:nn {
1622   \str_set:Nx \l_stex_import_archive_str { #1 }
1623   \str_set:Nn \l_stex_import_path_str { #2 }
1624
1625   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1626   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1627   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1628
1629   \stex_modules_current_namespace:
1630   \bool_lazy_all:nTF {
1631     {\str_if_empty_p:N \l_stex_import_archive_str}
1632     {\str_if_empty_p:N \l_stex_import_path_str}
1633     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1634   }{
1635     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1636     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1637   }{
1638     \str_if_empty:NT \l_stex_import_archive_str {
1639       \prop_if_exist:NT \l_stex_current_repository_prop {
1640         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1641       }
1642     }
1643     \str_if_empty:NTF \l_stex_import_archive_str {
1644       \str_if_empty:NF \l_stex_import_path_str {
1645         \str_set:Nx \l_stex_import_ns_str {
1646           \l_stex_module_ns_str / \l_stex_import_path_str
1647         }
1648       }
1649     }{
1650       \stex_require_repository:n \l_stex_import_archive_str
1651       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1652       \l_stex_import_ns_str
1653       \str_if_empty:NF \l_stex_import_path_str {
1654         \str_set:Nx \l_stex_import_ns_str {
1655           \l_stex_import_ns_str / \l_stex_import_path_str
1656         }
1657       }
1658     }
1659   }
1660 }
```

(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>

```
1664 \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\}}
1665 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1666   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1667
1668     % archive
1669     \str_set:Nx \l_tmpa_str { #2 }
1670     \str_if_empty:NTF \l_tmpa_str {
1671       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1672     } {
1673       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1674       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1675       \seq_put_right:Nn \l_tmpa_seq { source }
1676     }
1677
1678     % path
1679     \str_set:Nx \l_tmpb_str { #3 }
1680     \str_if_empty:NTF \l_tmpb_str {
1681       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1682
1683       \ltx@ifpackageloaded{babel} {
1684         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1685           { \language } \l_tmpb_str {
1686           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1687         }
1688       } {
1689         \str_clear:N \l_tmpb_str
1690       }
1691
1692       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1693       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1694         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1695       }{
1696         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1697         \IfFileExists{ \l_tmpa_str.tex }{
1698           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1699         }{
1700           % try english as default
1701           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1702           \IfFileExists{ \l_tmpa_str.en.tex }{
1703             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1704           }{
1705             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1706           }
1707         }
1708       }
1709
1710     } {
1711       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1712       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1713     }
```

```

1714 \ltx@ifpackageloaded{babel} {
1715   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1716     { \language } \l_tmpb_str {
1717       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1718     }
1719 } {
1720   \str_clear:N \l_tmpb_str
1721 }
1722
1723 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1724
1725 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1726 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1727   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1728 }{
1729   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1730   \IfFileExists{ \l_tmpa_str/#4.tex }{
1731     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1732   }{
1733     % try english as default
1734     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1735     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1736       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1737     }{
1738       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1739       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1740         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1741       }{
1742         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1743         \IfFileExists{ \l_tmpa_str.tex }{
1744           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1745         }{
1746           % try english as default
1747           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1748           \IfFileExists{ \l_tmpa_str.en.tex }{
1749             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1750           }{
1751             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1752           }
1753         }
1754       }
1755     }
1756   }
1757 }
1758 }
1759
1760 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1761   \seq_clear:N \l_stex_all_modules_seq
1762   \str_clear:N \l_stex_current_module_str
1763   \str_set:Nx \l_tmpb_str { #2 }
1764   \str_if_empty:NF \l_tmpb_str {
1765     \stex_set_current_repository:n { #2 }
1766   }
1767   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1768     }
1769
1770     \stex_if_module_exists:nF { #1 ? #4 } {
1771       \msg_error:nnx{stex}{error/unknownmodule}{
1772         #1?#4~(in~file~\g__stex_importmodule_file_str)
1773       }
1774     }
1775   }
1776   \stex_activate_module:n { #1 ? #4 }
1777 }

```

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

`\importmodule`

```

1778 \NewDocumentCommand \importmodule { 0{ } m } {
1779   \stex_import_module_uri:nn { #1 } { #2 }
1780   \stex_debug:nn{modules}{Importing~module:~
1781     \l_stex_import_ns_str ? \l_stex_import_name_str
1782   }
1783   \stex_if_smsmode:F {
1784     \stex_import_require_module:nnnn
1785     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1786     { \l_stex_import_path_str } { \l_stex_import_name_str }
1787     \stex_annotate_invisible:nnn
1788     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1789   }
1790   \exp_args:Nx \stex_add_to_current_module:n {
1791     \stex_import_require_module:nnnn
1792     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1793     { \l_stex_import_path_str } { \l_stex_import_name_str }
1794   }
1795   \exp_args:Nx \stex_add_import_to_current_module:n {
1796     \l_stex_import_ns_str ? \l_stex_import_name_str
1797   }
1798   \stex_smsmode_do:
1799   \ignorespacesandpars
1800 }
1801 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

1802 \NewDocumentCommand \usemodule { 0{ } m } {
1803   \stex_if_smsmode:F {
1804     \stex_import_module_uri:nn { #1 } { #2 }
1805     \stex_import_require_module:nnnn
1806     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1807     { \l_stex_import_path_str } { \l_stex_import_name_str }
1808     \stex_annotate_invisible:nnn
1809     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1810   }
1811   \stex_smsmode_do:
1812   \ignorespacesandpars
1813 }

```

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

1814 `\endpackage`

Chapter 30

STEX -Symbols Implementation

```
1815 <*package>
1816
1817 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1818
1819 \msg_new:nnn{stex}{error/wrongargs}{
1820   args~value~in~symbol~declaration~for~#1~
1821   needs~to~be~i,~a,~b~or~B,~but~#2~given
1822 }
```

30.1 Symbol Declarations

```
1823 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1824 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 35.)
```

\STEXsymbol

```
1825 \NewDocumentCommand \STEXsymbol { m } {
1826   \stex_get_symbol:n { #1 }
1827   \exp_args:No
1828   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1829 }
```

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

symdecl arguments:

```
1830 \keys_define:nn { stex / symdecl } {
1831   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1832   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1833   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1834   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1835   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1836   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
```

```

1837 gfc .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1838 specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1839 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1840 assoc .choices:nn =
1841 {bin,binl,binr,pre,conj,pwconj}
1842 {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1843 }
1844
1845 \bool_new:N \l_stex_symdecl_make_macro_bool
1846
1847 \cs_new_protected:Nn \__stex_symdecl_args:n {
1848 \str_clear:N \l_stex_symdecl_name_str
1849 \str_clear:N \l_stex_symdecl_args_str
1850 \str_clear:N \l_stex_symdecl_deprecate_str
1851 \str_clear:N \l_stex_symdecl_astype_str
1852 \bool_set_false:N \l_stex_symdecl_local_bool
1853 \tl_clear:N \l_stex_symdecl_type_tl
1854 \tl_clear:N \l_stex_symdecl_definiens_tl
1855
1856 \keys_set:nn { stex / symdecl } { #1 }
1857 }

```

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

```

1858
1859 \NewDocumentCommand \symdecl { s m O{}} {
1860 \__stex_symdecl_args:n { #3 }
1861 \IfBooleanTF #1 {
1862 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1863 } {
1864 \bool_set_true:N \l_stex_symdecl_make_macro_bool
1865 }
1866 \stex_symdecl_do:n { #2 }
1867 \stex_smsmode_do:
1868 }
1869
1870 \cs_new_protected:Nn \stex_symdecl_do:nn {
1871 \__stex_symdecl_args:n{#1}
1872 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1873 \stex_symdecl_do:n{#2}
1874 }
1875
1876 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1877 \cs_new_protected:Nn \stex_symdecl_do:n {
1878 \stex_if_in_module:F {
1879 % TODO throw error? some default namespace?
1880 }
1881
1882 \str_if_empty:NT \l_stex_symdecl_name_str {
1883 \str_set:Nx \l_stex_symdecl_name_str { #1 }

```

```

1884 }
1885
1886 \prop_if_exist:cT { \l_stex_symdecl_
1887   \l_stex_current_module_str ?
1888   \l_stex_symdecl_name_str
1889   _prop
1890 }{
1891   % TODO throw error (beware of circular dependencies)
1892 }
1893
1894 \prop_clear:N \l_tmpa_prop
1895 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1896 \seq_clear:N \l_tmpa_seq
1897 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1898 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1899
1900 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1901   \str_if_empty:NF \l_stex_module_deprecate_str {
1902     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1903   }
1904 }
1905 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1906
1907 \exp_args:No \stex_add_constant_to_current_module:n {
1908   \l_stex_symdecl_name_str
1909 }
1910
1911 % arity/args
1912 \int_zero:N \l_tmpb_int
1913
1914 \bool_set_true:N \l_tmpa_bool
1915 \str_map_inline:Nn \l_stex_symdecl_args_str {
1916   \token_case_meaning:NnF ##1 {
1917     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1918     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1919     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1920     {\tl_to_str:n a} {
1921       \bool_set_false:N \l_tmpa_bool
1922       \int_incr:N \l_tmpb_int
1923     }
1924     {\tl_to_str:n B} {
1925       \bool_set_false:N \l_tmpa_bool
1926       \int_incr:N \l_tmpb_int
1927     }
1928   }{
1929     \msg_error:nnxx{stex}{error/wrongargs}{
1930       \l_stex_current_module_str ?
1931       \l_stex_symdecl_name_str
1932     }{##1}
1933   }
1934 }
1935 \bool_if:NTF \l_tmpa_bool {
1936   % possibly numeric
1937   \str_if_empty:NTF \l_stex_symdecl_args_str {

```

```

1938     \prop_put:Nnn \l_tmpa_prop { args } {}
1939     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1940   }{
1941     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1942     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1943     \str_clear:N \l_tmpa_str
1944     \int_step_inline:nn \l_tmpa_int {
1945       \str_put_right:Nn \l_tmpa_str i
1946     }
1947     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1948   }
1949 } {
1950   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1951   \prop_put:Nnx \l_tmpa_prop { arity }
1952     { \str_count:N \l_stex_symdecl_args_str }
1953 }
1954 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1955
1956
1957 % semantic macro
1958
1959 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1960   \exp_args:Nx \stex_do_up_to_module:n {
1961     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1962       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1963     }}
1964   }
1965
1966   \bool_if:NF \l_stex_symdecl_local_bool {
1967     \exp_args:Nx \stex_add_to_current_module:n {
1968       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1969         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1970       } }
1971     }
1972   }
1973 }
1974
1975 % add to all symbols
1976
1977 \bool_if:NF \l_stex_symdecl_local_bool {
1978   \exp_args:Nx \stex_add_to_current_module:n {
1979     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1980       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1981     }
1982   }
1983   % \exp_args:Nx \stex_add_field_to_current_module:n {
1984   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1985   % }
1986 }
1987
1988 \stex_debug:nn{symbols}{New~symbol:~
1989   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1990   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1991   Args:~\prop_item:Nn \l_tmpa_prop { args }

```

```

1992 }
1993
1994 % circular dependencies require this:
1995
1996 \prop_if_exist:cF {
1997   l_stex_symdecl_
1998   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1999   _prop
2000 } {
2001   \prop_set_eq:cN {
2002     l_stex_symdecl_
2003     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2004     _prop
2005   } \l_tmpa_prop
2006 }
2007
2008 \seq_clear:c {
2009   l_stex_symdecl_
2010   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2011   _notations
2012 }
2013
2014 \bool_if:NF \l_stex_symdecl_local_bool {
2015   \exp_args:Nx
2016   \stex_add_to_current_module:n {
2017     \seq_clear:c {
2018       l_stex_symdecl_
2019       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020       _notations
2021     }
2022     \prop_set_from_keyval:cn {
2023       l_stex_symdecl_
2024       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2025       _prop
2026     } {
2027       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2028       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2029       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2030       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2031       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2032       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2033     }
2034   }
2035 }
2036
2037 \stex_if_smsmode:F {
2038   \exp_args:Nx \stex_do_up_to_module:n {
2039     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2040       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041     }
2042   }
2043   \stex_if_do_html:T {
2044     \stex_annotate_invisible:nnn {symdecl} {
2045       \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2046     } {
2047       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}}{${\l_st
2048       \stex_annotate_invisible:nnn{args}}{ }{
2049         \prop_item:Nn \l_tmpa_prop { args }
2050       }
2051       \stex_annotate_invisible:nnn{macroname}{#1}{ }
2052       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2053         \stex_annotate_invisible:nnn{definiens}{ }
2054         {${\l_stex_symdecl_definiens_tl$}
2055       }
2056       \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2057         \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2058       }
2059     }
2060   }
2061 }
2062 }

```

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

`\stex_get_symbol:n`

```

2063 \str_new:N \l_stex_get_symbol_uri_str
2064
2065 \cs_new_protected:Nn \stex_get_symbol:n {
2066   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2067     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2068   }{
2069     % argument is a string
2070     % is it a command name?
2071     \cs_if_exist:cTF { #1 }{
2072       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2073       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2074       \str_if_empty:NTF \l_tmpa_str {
2075         \exp_args:Nx \cs_if_eq:NNTF {
2076           \tl_head:N \l_tmpa_tl
2077         } \stex_invoke_symbol:n {
2078           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2079         }{
2080           \__stex_symdecl_get_symbol_from_string:n { #1 }
2081         }
2082       } {
2083         \__stex_symdecl_get_symbol_from_string:n { #1 }
2084       }
2085     }{
2086       % argument is not a command name
2087       \__stex_symdecl_get_symbol_from_string:n { #1 }
2088       % \l_stex_all_symbols_seq
2089     }
2090   }
2091   \str_if_eq:eeF {
2092     \prop_item:cn {
2093       \l_stex_symdecl \l_stex_get_symbol_uri_str _prop
2094     }{ deprecate }
2095   }{ }

```

```

2096 \msg_warning:nnxx{stex}{warning/deprecated}{
2097   Symbol~\l_stex_get_symbol_uri_str
2098 }{
2099   \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str _prop}{ deprecate }
2100 }
2101 }
2102 }
2103
2104 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2105   \str_set:Nn \l_tmpa_str { #1 }
2106   \bool_set_false:N \l_tmpa_bool
2107   \stex_if_in_module:T {
2108     \exp_args:Nno \seq_if_in:cnT {c_stex_module_l_stex_current_module_str _constants} { \l_
2109       \bool_set_true:N \l_tmpa_bool
2110       \str_set:Nx \l_stex_get_symbol_uri_str {
2111         \l_stex_current_module_str ? #1
2112       }
2113     }
2114   }
2115   \bool_if:NF \l_tmpa_bool {
2116     \tl_set:Nn \l_tmpa_tl {
2117       \msg_set:nnn{stex}{error/unknownsymbol}{
2118         No~symbol~#1~found!
2119       }
2120       \msg_error:nn{stex}{error/unknownsymbol}
2121     }
2122     \str_set:Nn \l_tmpa_str { #1 }
2123     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2124     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2125       \str_set:Nn \l_tmpb_str { ##1 }
2126       \str_if_eq:eeT { \l_tmpa_str } {
2127         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2128       } {
2129         \seq_map_break:n {
2130           \tl_set:Nn \l_tmpa_tl {
2131             \str_set:Nn \l_stex_get_symbol_uri_str {
2132               ##1
2133             }
2134           }
2135         }
2136       }
2137     }
2138     \l_tmpa_tl
2139   }
2140 }
2141
2142 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2143   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2144     { \tl_tail:N \l_tmpa_tl }
2145   \tl_if_single:NTF \l_tmpa_tl {
2146     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2147       \exp_after:wN \str_set:Nn \exp_after:wN
2148         \l_stex_get_symbol_uri_str \l_tmpa_tl
2149     }{

```

```

2150      % TODO
2151      % tail is not a single group
2152    }
2153  }{
2154    % TODO
2155    % tail is not a single group
2156  }
2157 }

```

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

30.2 Notations

```

2158 <@@=stex_notation>
      notation arguments:
2159 \keys_define:nn { stex / notation } {
2160   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2161   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2162   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2163   op        .tl_set:N   = \l__stex_notation_op_tl ,
2164   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2165   primary   .default:n  = {true} ,
2166   unknown   .code:n     = \str_set:Nx
2167               \l__stex_notation_variant_str \l_keys_key_str
2168 }
2169
2170 \cs_new_protected:Nn \_stex_notation_args:n {
2171   \str_clear:N \l__stex_notation_lang_str
2172   \str_clear:N \l__stex_notation_variant_str
2173   \str_clear:N \l__stex_notation_prec_str
2174   \tl_clear:N \l__stex_notation_op_tl
2175   \bool_set_false:N \l__stex_notation_primary_bool
2176
2177   \keys_set:nn { stex / notation } { #1 }
2178 }

```

\notation

```

2179 \NewDocumentCommand \notation { s m O{} } {
2180   \_stex_notation_args:n { #3 }
2181   \tl_clear:N \l_stex_symdecl_definiens_tl
2182   \stex_get_symbol:n { #2 }
2183   \tl_set:Nn \l_stex_notation_after_do_tl {
2184     \__stex_notation_final:
2185     \IfBooleanTF#1{
2186       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2187     }{}
2188     \stex_smsmode_do:
2189   }
2190   \stex_notation_do:nnnn
2191   { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { args } }
2192   { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str_prop } { arity } }
2193   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2194 }
2195 \stex_deactivate_macro:Nn \notation {module~environments}

```


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

\stex_notation_do:nnnn

```

2196 \seq_new:N \l__stex_notation_precedences_seq
2197 \tl_new:N \l__stex_notation_opprec_tl
2198 \int_new:N \l__stex_notation_currarg_int
2199 \tl_new:N \stex_symbol_after_invocation_tl
2200
2201 \cs_new_protected:Nn \stex_notation_do:nnnn {
2202   \let\l__stex_current_symbol_str\relax
2203   \seq_clear:N \l__stex_notation_precedences_seq
2204   \tl_clear:N \l__stex_notation_opprec_tl
2205   \str_set:Nx \l__stex_notation_args_str { #1 }
2206   \str_set:Nx \l__stex_notation_arity_str { #2 }
2207   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2208
2209   % precedences
2210   \str_if_empty:NTF \l__stex_notation_prec_str {
2211     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2212       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2213     }{
2214       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2215     }
2216   } {
2217     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2218       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2219       \int_step_inline:nn { \l__stex_notation_arity_str } {
2220         \exp_args:NNo
2221         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2222       }
2223     }{
2224       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2225       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2226         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2227         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2228           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2229           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2230           \seq_map_inline:Nn \l_tmpa_seq {
2231             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2232           }
2233         }
2234       }{
2235         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2236           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2237         }{
2238           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2239         }
2240       }
2241     }
2242   }
2243
2244   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2245   \int_step_inline:nn { \l__stex_notation_arity_str } {
2246     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {

```

```

2247     \exp_args:NNo
2248     \seq_put_right:No \l__stex_notation_precedences_seq {
2249         \l__stex_notation_opprec_tl
2250     }
2251 }
2252 }
2253 \tl_clear:N \l_stex_notation_dummyargs_tl
2254
2255 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2256     \exp_args:NNe
2257     \cs_set:Npn \l_stex_notation_macrocode_cs {
2258         \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2259         { \_stex_notation_suffix_str }
2260         { \l__stex_notation_opprec_tl }
2261         { \exp_not:n { #4 } }
2262     }
2263     \l_stex_notation_after_do_tl
2264 }{
2265     \str_if_in:NnTF \l__stex_notation_args_str b {
2266         \exp_args:Nne \use:nn
2267         {
2268             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2269             \cs_set:Npn \l__stex_notation_arity_str } { {
2270                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2271                 { \_stex_notation_suffix_str }
2272                 { \l__stex_notation_opprec_tl }
2273                 { \exp_not:n { #4 } }
2274             } }
2275 }{
2276     \str_if_in:NnTF \l__stex_notation_args_str B {
2277         \exp_args:Nne \use:nn
2278         {
2279             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2280             \cs_set:Npn \l__stex_notation_arity_str } { {
2281                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2282                 { \_stex_notation_suffix_str }
2283                 { \l__stex_notation_opprec_tl }
2284                 { \exp_not:n { #4 } }
2285             } }
2286 }{
2287     \exp_args:Nne \use:nn
2288     {
2289         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2290         \cs_set:Npn \l__stex_notation_arity_str } { {
2291             \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2292             { \_stex_notation_suffix_str }
2293             { \l__stex_notation_opprec_tl }
2294             { \exp_not:n { #4 } }
2295         } }
2296     }
2297 }
2298
2299 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2300 \int_zero:N \l__stex_notation_currarg_int

```

```

2301     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2302     \__stex_notation_arguments:
2303   }
2304 }

```

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

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

```

2305 \cs_new_protected:Nn \__stex_notation_arguments: {
2306   \int_incr:N \l__stex_notation_currarg_int
2307   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2308     \l_stex_notation_after_do_tl
2309   }{
2310     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2311     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2312     \str_if_eq:VnTF \l_tmpa_str a {
2313       \__stex_notation_argument_assoc:n
2314     }{
2315       \str_if_eq:VnTF \l_tmpa_str B {
2316         \__stex_notation_argument_assoc:n
2317       }{
2318         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2319         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2320           { \stex_term_math_arg:nnn
2321             { \int_use:N \l__stex_notation_currarg_int }
2322             { \l_tmpa_str }
2323             { ####\int_use:N \l__stex_notation_currarg_int }
2324           }
2325         }
2326         \__stex_notation_arguments:
2327       }
2328     }
2329   }
2330 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2331 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2332
2333   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2334     {\l__stex_notation_arity_str}{
2335     #1
2336   }
2337   \int_zero:N \l_tmpa_int
2338   \tl_clear:N \l_tmpa_tl
2339   \str_map_inline:Nn \l__stex_notation_args_str {
2340     \int_incr:N \l_tmpa_int
2341     \tl_put_right:Nx \l_tmpa_tl {
2342       \str_if_eq:nnTF {##1}{a}{ {} }{
2343         \str_if_eq:nnTF {##1}{B}{ {} }{
2344           {\stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2345         }
2346     }

```

```

2347     }
2348   }
2349   \exp_after:wN\exp_after:wN\exp_after:wN \def
2350   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2351   \exp_after:wN\exp_after:wN\exp_after:wN ##
2352   \exp_after:wN\exp_after:wN\exp_after:wN 1
2353   \exp_after:wN\exp_after:wN\exp_after:wN ##
2354   \exp_after:wN\exp_after:wN\exp_after:wN 2
2355   \exp_after:wN\exp_after:wN\exp_after:wN {
2356     \exp_after:wN \exp_after:wN \exp_after:wN
2357     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2358       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2359     }
2360   }
2361
2362   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2363   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2364     \stex_term_math_assoc_arg:nnnn
2365     { \int_use:N \l__stex_notation_currarg_int }
2366     { \l_tmpa_str }
2367     { ####\int_use:N \l__stex_notation_currarg_int }
2368     { \l_tmpa_cs {####1} {####2} }
2369   } }
2370   \__stex_notation_arguments:
2371 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2372 \cs_new_protected:Nn \__stex_notation_final: {
2373   \exp_args:Nne \use:nn
2374   {
2375     \cs_generate_from_arg_count:cNnn {
2376       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2377       \__stex_notation_suffix_str
2378       _cs
2379     }
2380     \cs_set:Npn \l__stex_notation_arity_str { { {
2381       \exp_after:wN \exp_after:wN \exp_after:wN
2382       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2383       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2384     } } }
2385
2386     \tl_if_empty:NF \l__stex_notation_op_tl {
2387       \cs_set:cpx {
2388         stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2389         \__stex_notation_suffix_str
2390         _cs
2391       } {
2392         \stex_term_oms:nnn {
2393           \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2394         }{
2395           \l_stex_get_symbol_uri_str
2396         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }

```

```

2397     }
2398 }
2399
2400 \exp_args:Ne
2401 \stex_add_to_current_module:n {
2402   \cs_generate_from_arg_count:cNnn {
2403     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2404     \__stex_notation_suffix_str
2405     _cs
2406   } \cs_set:Npn {\l__stex_notation_arity_str} {
2407     \exp_after:wN \exp_after:wN \exp_after:wN
2408     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2409     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2410   }
2411   \tl_if_empty:NF \l__stex_notation_op_tl {
2412     \cs_set:cpn {
2413       stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2414       \__stex_notation_suffix_str
2415       _cs
2416     } {
2417       \_stex_term_oms:nnn {
2418         \l_stex_get_symbol_uri_str \c_hash_str \__stex_notation_suffix_str
2419       } {
2420         \l_stex_get_symbol_uri_str
2421       } { \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } } }
2422     }
2423   }
2424 }
2425 %\exp_args:Nx
2426 % \stex_do_up_to_module:n {
2427   \seq_put_right:cx {
2428     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2429     _notations
2430   } {
2431     \__stex_notation_suffix_str
2432   }
2433 % }
2434
2435 \stex_debug:nn{symbols}{
2436   Notation~\__stex_notation_suffix_str
2437   ~for~\l_stex_get_symbol_uri_str^^J
2438   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2439   Argument~precedences:~
2440   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2441   Notation: \cs_meaning:c {
2442     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2443     \__stex_notation_suffix_str
2444     _cs
2445   }
2446 }
2447
2448 \exp_args:Ne
2449 \stex_add_to_current_module:n {
2450   \seq_put_right:cn {

```

```

2451     l_stex_symdecl \l_stex_get_symbol_uri_str
2452     _notations
2453   } { \_stex_notation_suffix_str }
2454 }
2455
2456 \stex_if_smsmode:F {
2457
2458   % HTML annotations
2459   \stex_if_do_html:T {
2460     \stex_annotate_invisible:nnn { notation }
2461     { \l_stex_get_symbol_uri_str } {
2462       \stex_annotate_invisible:nnn { notationfragment }
2463       { \_stex_notation_suffix_str }{}
2464       \stex_annotate_invisible:nnn { precedence }
2465       { \l__stex_notation_prec_str }{}
2466
2467       \int_zero:N \l_tmpa_int
2468       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2469       \tl_clear:N \l_tmpa_tl
2470       \int_step_inline:nn { \l__stex_notation_arity_str }{
2471         \int_incr:N \l_tmpa_int
2472         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2473         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2474         \str_if_eq:VnTF \l_tmpb_str a {
2475           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2476             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2477             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2478           } }
2479         }{
2480           \str_if_eq:VnTF \l_tmpb_str B {
2481             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2482               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2483               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2484             } }
2485           }{
2486             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2487               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2488             } }
2489           }
2490         }
2491       }
2492       \stex_annotate_invisible:nnn { notationcomp }{}{
2493         \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2494         $ \exp_args:Nno \use:nn { \use:c {
2495           stex_notation_ \l_stex_current_symbol_str
2496           \c_hash_str \_stex_notation_suffix_str _cs
2497         } } { \l_tmpa_tl } $
2498       }
2499     }
2500   }
2501 }
2502 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2503 \keys_define:nn { stex / setnotation } {
2504   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2505   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2506   unknown .code:n = \str_set:Nx
2507     \l__stex_notation_variant_str \l_keys_key_str
2508 }
2509
2510 \cs_new_protected:Nn \stex_setnotation_args:n {
2511   \str_clear:N \l__stex_notation_lang_str
2512   \str_clear:N \l__stex_notation_variant_str
2513   \keys_set:nn { stex / setnotation } { #1 }
2514 }
2515
2516 \cs_new_protected:Nn \stex_setnotation:n {
2517   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2518     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2519     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2520       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2521     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2522       { \c_hash_str }
2523     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2524       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2525     \exp_args:Nx \stex_add_to_current_module:n {
2526       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2527         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2528       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2529         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2530       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2531         { \c_hash_str }
2532     }
2533     \stex_debug:nn {notations}{
2534       Setting~default~notation~
2535       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2536       #1 \\
2537       \expandafter\meaning\csname
2538         l_stex_symdecl_#1 _notations\endcsname
2539     }
2540   }{
2541     % todo throw error
2542   }
2543 }
2544
2545 \NewDocumentCommand \setnotation {m m} {
2546   \stex_get_symbol:n { #1 }
2547   \stex_setnotation_args:n { #2 }
2548   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2549   \stex_smsmode_do:
2550 }
2551
2552 \cs_new_protected:Nn \stex_copy_notations:nn {
2553   \stex_debug:nn {notations}{
2554     Copying~notations~from~#2~to~#1\\
2555     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```

```

2556 }
2557 \tl_clear:N \l_tmpa_tl
2558 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2559   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2560 }
2561 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2562   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2563   \edef \l_tmpa_tl {
2564     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2565     \exp_after:wN\exp_after:wN\exp_after:wN {
2566       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2567     }
2568   }
2569   \exp_args:Nx
2570   \stex_do_up_to_module:n {
2571     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2572     \cs_generate_from_arg_count:cNnn {
2573       stex_notation_ #1 \c_hash_str ##1 _cs
2574     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2575       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2576     }
2577   }
2578 }
2579 }
2580
2581 \NewDocumentCommand \copynotation {m m} {
2582   \stex_get_symbol:n { #1 }
2583   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2584   \stex_get_symbol:n { #2 }
2585   \exp_args:Noo
2586   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2587   \exp_args:Nx \stex_add_import_to_current_module:n{
2588     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2589   }
2590   \stex_smsmode_do:
2591 }
2592

```

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

\symdef

```

2593 \keys_define:nn { stex / symdef } {
2594   name .str_set_x:N = \l_stex_symdecl_name_str ,
2595   local .bool_set:N = \l_stex_symdecl_local_bool ,
2596   args .str_set_x:N = \l_stex_symdecl_args_str ,
2597   type .tl_set:N = \l_stex_symdecl_type_tl ,
2598   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2599   op .tl_set:N = \l_stex_notation_op_tl ,
2600   lang .str_set_x:N = \l_stex_notation_lang_str ,
2601   variant .str_set_x:N = \l_stex_notation_variant_str ,
2602   prec .str_set_x:N = \l_stex_notation_prec_str ,
2603   assoc .choices:nn =
2604     {bin,binl,binr,pre,conj,pwconj}
2605     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},

```



```

2606   unknown .code:n      = \str_set:Nx
2607       \l__stex_notation_variant_str \l_keys_key_str
2608 }
2609
2610 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2611   \str_clear:N \l_stex_symdecl_name_str
2612   \str_clear:N \l_stex_symdecl_args_str
2613   \str_clear:N \l_stex_symdecl_assoctype_str
2614   \bool_set_false:N \l_stex_symdecl_local_bool
2615   \tl_clear:N \l_stex_symdecl_type_tl
2616   \tl_clear:N \l_stex_symdecl_definiens_tl
2617   \str_clear:N \l__stex_notation_lang_str
2618   \str_clear:N \l__stex_notation_variant_str
2619   \str_clear:N \l__stex_notation_prec_str
2620   \tl_clear:N \l__stex_notation_op_tl
2621
2622   \keys_set:nn { stex / symdef } { #1 }
2623 }
2624
2625 \NewDocumentCommand \symdef { m O{} } {
2626   \__stex_notation_symdef_args:n { #2 }
2627   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2628   \stex_symdecl_do:n { #1 }
2629   \tl_set:Nn \l_stex_notation_after_do_tl {
2630     \__stex_notation_final:
2631     \stex_smsmode_do:
2632   }
2633   \str_set:Nx \l_stex_get_symbol_uri_str {
2634     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2635   }
2636   \exp_args:Nx \stex_notation_do:nnnn
2637     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { args } }
2638     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { arity } }
2639     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2640   }
2641   \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

30.3 Variables

```

2642 <@@=stex_variables>
2643
2644 \keys_define:nn { stex / vardef } {
2645   name .str_set_x:N = \l__stex_variables_name_str ,
2646   args .str_set_x:N = \l__stex_variables_args_str ,
2647   type .tl_set:N    = \l__stex_variables_type_tl ,
2648   def  .tl_set:N    = \l__stex_variables_def_tl ,
2649   op   .tl_set:N    = \l__stex_variables_op_tl ,
2650   prec .str_set_x:N = \l__stex_variables_prec_str ,
2651   assoc .choices:nn =
2652     {bin,binl,binr,pre,conj,pwconj}
2653     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2654   bind .choices:nn =

```

```

2655     {forall,exists}
2656     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2657   }
2658
2659   \cs_new_protected:Nn \__stex_variables_args:n {
2660     \str_clear:N \l__stex_variables_name_str
2661     \str_clear:N \l__stex_variables_args_str
2662     \str_clear:N \l__stex_variables_prec_str
2663     \str_clear:N \l__stex_variables_assoctype_str
2664     \str_clear:N \l__stex_variables_bind_str
2665     \tl_clear:N \l__stex_variables_type_tl
2666     \tl_clear:N \l__stex_variables_def_tl
2667     \tl_clear:N \l__stex_variables_op_tl
2668
2669     \keys_set:nn { stex / vardef } { #1 }
2670   }
2671
2672   \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2673     \__stex_variables_args:n {#2}
2674     \str_if_empty:NT \l__stex_variables_name_str {
2675       \str_set:Nx \l__stex_variables_name_str { #1 }
2676     }
2677     \prop_clear:N \l_tmpa_prop
2678     \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2679
2680     \int_zero:N \l_tmpb_int
2681     \bool_set_true:N \l_tmpa_bool
2682     \str_map_inline:Nn \l__stex_variables_args_str {
2683       \token_case_meaning:NnF ##1 {
2684         0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2685         {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2686         {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2687         {\tl_to_str:n a} {
2688           \bool_set_false:N \l_tmpa_bool
2689           \int_incr:N \l_tmpb_int
2690         }
2691         {\tl_to_str:n B} {
2692           \bool_set_false:N \l_tmpa_bool
2693           \int_incr:N \l_tmpb_int
2694         }
2695       }{
2696         \msg_error:nnxx{stex}{error/wrongargs}{
2697           variable~\l__stex_variables_name_str
2698         }{##1}
2699       }
2700     }
2701     \bool_if:NTF \l_tmpa_bool {
2702       % possibly numeric
2703       \str_if_empty:NTF \l__stex_variables_args_str {
2704         \prop_put:Nnn \l_tmpa_prop { args } {}
2705         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2706       }{
2707         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2708         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

2709     \str_clear:N \l_tmpa_str
2710     \int_step_inline:nn \l_tmpa_int {
2711       \str_put_right:Nn \l_tmpa_str i
2712     }
2713     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2714     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2715   }
2716 } {
2717   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2718   \prop_put:Nnx \l_tmpa_prop { arity }
2719   { \str_count:N \l__stex_variables_args_str }
2720 }
2721 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2722 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2723
2724 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2725
2726 \tl_if_empty:NF \l__stex_variables_op_tl {
2727   \cs_set:cpx {
2728     stex_var_op_notation_ \l__stex_variables_name_str _cs
2729   } {
2730     \stex_term_omv:nn {
2731       var://\l__stex_variables_name_str
2732     } { \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2733   }
2734 }
2735
2736 \tl_set:Nn \l_stex_notation_after_do_tl {
2737   \exp_args:Nne \use:nn {
2738     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2739     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2740   } { {
2741     \exp_after:wN \exp_after:wN \exp_after:wN
2742     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2743     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2744   } }
2745   \stex_if_do_html:T {
2746     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2747       \stex_annotate_invisible:nnn { precedence }
2748       { \l__stex_variables_prec_str } { }
2749     \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $ \l
2750     \stex_annotate_invisible:nnn { args } { } { \l__stex_variables_args_str }
2751     \stex_annotate_invisible:nnn { macroname } { #1 } { }
2752     \tl_if_empty:NF \l__stex_variables_def_tl {
2753       \stex_annotate_invisible:nnn { definiens } { }
2754       { $ \l__stex_variables_def_tl $ }
2755     }
2756     \str_if_empty:NF \l__stex_variables_assoc_type_str {
2757       \stex_annotate_invisible:nnn { assoc_type } { \l__stex_variables_assoc_type_str } { }
2758     }
2759     \int_zero:N \l_tmpa_int
2760     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2761     \tl_clear:N \l_tmpa_tl
2762     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {

```

```

2763 \int_incr:N \l_tmpa_int
2764 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2765 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2766 \str_if_eq:VnTF \l_tmpb_str a {
2767 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2768 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2769 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2770 } }
2771 }{
2772 \str_if_eq:VnTF \l_tmpb_str B {
2773 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2774 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2775 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2776 } }
2777 }{
2778 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2779 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2780 } }
2781 }
2782 }
2783 }
2784 \stex_annotate_invisible:nnn { notationcomp }{}{
2785 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2786 $ \exp_args:Nno \use:nn { \use:c {
2787 stex_var_notation_\l__stex_variables_name_str_cs
2788 } } { \l_tmpa_tl } $
2789 }
2790 }
2791 }
2792 }
2793
2794 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { arit
2795 }
2796
2797 \cs_new:Nn \__stex_variables_reset:N {
2798 \tl_if_exist:NTF #1 {
2799 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2800 }{
2801 \let \exp_not:N #1 \exp_not:N \undefined
2802 }
2803 }
2804
2805 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2806 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2807 \exp_args:Nnx \use:nn {
2808 % TODO
2809 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2810 #2
2811 }
2812 }{
2813 \__stex_variables_reset:N \varnot
2814 \__stex_variables_reset:N \vartype
2815 \__stex_variables_reset:N \vardef
2816 }

```

```

2817 }
2818
2819 \NewDocumentCommand \vardef { s } {
2820   \IfBooleanTF#1 {
2821     \__stex_variables_do_complex:nn
2822   }{
2823     \__stex_variables_do_simple:nnn
2824   }
2825 }
2826
2827 \NewDocumentCommand \svar { 0{} m }{
2828   \tl_if_empty:nTF {#1}{
2829     \str_set:Nn \l_tmpa_str { #2 }
2830   }{
2831     \str_set:Nn \l_tmpa_str { #1 }
2832   }
2833   \_stex_term_omv:nn {
2834     var://\l_tmpa_str
2835   }{ \comp{ #2 } }
2836 }
2837
2838 </package>

```

Chapter 31

STEX -Terms Implementation

```
2839 <*package>
2840
2841 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2842
2843 <@@=stex_terms>
2844
2845   Warnings and error messages
2846 \msg_new:nnn{stex}{error/nonotation}{
2847   Symbol~#1~invoked,~but~has~no~notation~#2!
2848 }
2849 \msg_new:nnn{stex}{error/notationarg}{
2850   Error~in~parsing~notation~#1
2851 }
2852 \msg_new:nnn{stex}{error/noop}{
2853   Symbol~#1~has~no~operator~notation~for~notation~#2
2854 }
2855 \msg_new:nnn{stex}{error/notallowed}{
2856   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2857 }
```

31.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2857 \keys_define:nn { stex / terms } {
2858   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2859   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2860   unknown .code:n = \str_set:Nx
2861     \l__stex_terms_variant_str \l_keys_key_str
2862 }
2863
2864 \cs_new_protected:Nn \__stex_terms_args:n {
2865   \str_clear:N \l__stex_terms_lang_str
2866   \str_clear:N \l__stex_terms_variant_str
2867 }
```

```

2868 \keys_set:nn { stex / terms } { #1 }
2869 }
2870
2871 \cs_new:Nn \__stex_terms_reset:N {
2872   \tl_if_exist:NTF #1 {
2873     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2874   }{
2875     \let \exp_not:N #1 \exp_not:N \undefined
2876   }
2877 }
2878
2879 \bool_new:N \l__stex_terms_allow_semantic_bool
2880 \bool_set_true:N \l__stex_terms_allow_semantic_bool
2881
2882 \cs_new_protected:Nn \stex_invoke_symbol:n {
2883   \bool_if:NTF \l__stex_terms_allow_semantic_bool {
2884     \str_if_eq:eeF {
2885       \prop_item:cn {
2886         l_stex_symdecl_#1_prop
2887       }{ deprecate }
2888     }{}{
2889       \msg_warning:nxxx{stex}{warning/deprecated}{
2890         Symbol~#1
2891       }{
2892         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2893       }
2894     }
2895     \if_mode_math:
2896       \exp_after:wN \__stex_terms_invoke_math:n
2897     \else:
2898       \exp_after:wN \__stex_terms_invoke_text:n
2899     \fi: { #1 }
2900   }{
2901     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2902   }
2903 }
2904
2905 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2906   \peek_charcode_remove:NTF ! {
2907     \__stex_terms_invoke_op_custom:nn {#1}
2908   }{
2909     \__stex_terms_invoke_custom:nn {#1}
2910   }
2911 }
2912
2913 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2914   \peek_charcode_remove:NTF ! {
2915     % operator
2916     \peek_charcode_remove:NTF * {
2917       % custom op
2918       \__stex_terms_invoke_op_custom:nn {#1}
2919     }{
2920       % op notation
2921       \peek_charcode:NTF [ {

```

```

2922     \__stex_terms_invoke_op_notation:nw {#1}
2923   }{
2924     \__stex_terms_invoke_op_notation:nw {#1}[]
2925   }
2926 }
2927 }{
2928   \peek_charcode_remove:NTF * {
2929     \__stex_terms_invoke_custom:nn {#1}
2930     % custom
2931   }{
2932     % normal
2933     \peek_charcode:NTF [ {
2934       \__stex_terms_invoke_notation:nw {#1}
2935     }{
2936       \__stex_terms_invoke_notation:nw {#1}[]
2937     }
2938   }
2939 }
2940 }
2941
2942
2943 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
2944   \exp_args:Nnx \use:nn {
2945     \str_set:Nn \l_stex_current_symbol_str { #1 }
2946     \bool_set_false:N \l__stex_terms_allow_semantic_bool
2947     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2948       \comp{ #2 }
2949     }
2950   }{
2951     \__stex_terms_reset:N \l_stex_current_symbol_str
2952     \bool_set_true:N \l__stex_terms_allow_semantic_bool
2953   }
2954 }
2955
2956 \cs_new_protected:Nn \__stex_terms_find_notation:nn {
2957   \str_set:Nn \l_stex_current_symbol_str { #1 }
2958   \__stex_terms_args:n { #2 }
2959   \seq_if_empty:cTF {
2960     l_stex_symdecl_ #1 _notations
2961   } {
2962     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2963   } {
2964     \bool_lazy_all:nTF {
2965       {\str_if_empty_p:N \l__stex_terms_variant_str}
2966       {\str_if_empty_p:N \l__stex_terms_lang_str}
2967     }{
2968       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l__stex_terms_variant_str
2969     }{
2970       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
2971         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2972       }{
2973         \str_set:Nx \l__stex_terms_variant_str { \l__stex_terms_variant_str \c_hash_str \l__
2974       }{
2975         \msg_error:nnxx{stex}{error/nonotation}{#1}{

```



```

2976         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2977     }
2978 }
2979 }
2980 }
2981 }
2982
2983 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
2984     \__stex_terms_find_notation:nn { #1 }{ #2 }
2985     \bool_set_false:N \l__stex_terms_allow_semantic_bool
2986     \cs_if_exist:cTF {
2987         stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
2988     }{
2989         \use:c{stex_op_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
2990     }{
2991         \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str}
2992     }
2993     \bool_set_true:N \l__stex_terms_allow_semantic_bool
2994 }
2995
2996 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
2997     \__stex_terms_find_notation:nn { #1 }{ #2 }
2998     \cs_if_exist:cTF {
2999         stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs
3000     }{
3001         \tl_set:Nx \stex_symbol_after_invocation_tl {
3002             \__stex_terms_reset:N \stex_symbol_after_invocation_tl
3003             \__stex_terms_reset:N \l_stex_current_symbol_str
3004             \bool_set_true:N \l__stex_terms_allow_semantic_bool
3005         }
3006         \bool_set_false:N \l__stex_terms_allow_semantic_bool
3007         \use:c{stex_notation_ #1 \c_hash_str \l__stex_terms_variant_str _cs}
3008     }{
3009         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3010             ~\l__stex_terms_variant_str
3011         }
3012     }
3013 }
3014
3015 \prop_new:N \l__stex_terms_custom_args_prop
3016
3017 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3018     \exp_args:Nnx \use:nn {
3019         \bool_set_false:N \l__stex_terms_allow_semantic_bool
3020         \str_set:Nn \l_stex_current_symbol_str { #1 }
3021         \prop_clear:N \l__stex_terms_custom_args_prop
3022         \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3023         \prop_get:cnN {
3024             l_stex_symdecl_#1 _prop
3025         }{ args } \l_tmpa_str
3026         \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3027         \tl_set:Nn \arg { \__stex_terms_arg: }
3028         \str_if_empty:NTF \l_tmpa_str {
3029             \_stex_term_oms:nnn {#1}{#1}{#2}

```

```

3030     }{
3031         \str_if_in:NnTF \l_tmpa_str b {
3032             \stex_term_ombind:nnn {#1}{#1}{#2}
3033         }{
3034             \str_if_in:NnTF \l_tmpa_str B {
3035                 \stex_term_ombind:nnn {#1}{#1}{#2}
3036             }{
3037                 \stex_term_oma:nnn {#1}{#1}{#2}
3038             }
3039         }
3040     }
3041     % TODO check that all arguments exist
3042 }{
3043     \__stex_terms_reset:N \l_stex_current_symbol_str
3044     \__stex_terms_reset:N \arg
3045     \__stex_terms_reset:N \l__stex_terms_custom_args_prop
3046     \bool_set_true:N \l__stex_terms_allow_semantic_bool
3047 }
3048 }
3049
3050 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3051     \tl_if_empty:nTF {#2}{
3052         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3053         \bool_set_true:N \l_tmpa_bool
3054         \bool_do_while:Nn \l_tmpa_bool {
3055             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3056             \int_incr:N \l_tmpa_int
3057         }{
3058             \bool_set_false:N \l_tmpa_bool
3059         }
3060     }
3061 }{
3062     \int_set:Nn \l_tmpa_int { #2 }
3063     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3064         % TODO throw error
3065     }
3066 }
3067 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3068 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3069     % TODO throw error
3070 }
3071 \IfBooleanTF#1{
3072     \stex_annotate_invisible:n {
3073         \exp_args:No \stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3074     }
3075 }{
3076     \exp_args:No \stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3077 }
3078 }
3079
3080
3081 \cs_new_protected:Nn \stex_term_arg:nn {
3082     \exp_args:Nnx \use:nn {
3083         \bool_set_true:N \l__stex_terms_allow_semantic_bool

```

```

3084 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3085 }{
3086 \bool_set_false:N \l__stex_terms_allow_semantic_bool
3087 }
3088 }
3089
3090 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3091 \exp_args:Nnx \use:nn
3092 { \int_set:Nn \l__stex_terms_downprec { #2 }
3093 \_stex_term_arg:nn { #1 }{ #3 }
3094 }
3095 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3096 }
3097
3098

```

(End definition for `\stex_invoke_symbol:n`. 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_math_assoc_arg:nnnn`

```

3195 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3196   % TODO sequences
3197   \clist_set:Nn \l_tmpa_clist{ #3 }
3198   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3199     \tl_set:Nn \l_tmpa_tl { #3 }
3200   }{
3201     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3202     \clist_reverse:N \l_tmpa_clist
3203     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3204
3205     \clist_map_inline:Nn \l_tmpa_clist {
3206       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3207         \exp_args:Nno
3208         \l_tmpa_cs { ##1 } \l_tmpa_tl
3209       }
3210     }
3211   }
3212   \exp_args:Nnno
3213   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3214 }

```

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

`\stex_term_custom:nn`

```

3215 \cs_new_protected:Nn \stex_term_custom:nn {
3216   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3217   \str_set:Nn \l_tmpa_str { #2 }
3218   \tl_clear:N \l_tmpa_tl
3219   \int_zero:N \l_tmpa_int
3220   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3221   \__stex_terms_custom_loop:
3222 }

```

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

`__stex_terms_custom_loop:`

```

3223 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3224   \bool_set_false:N \l_tmpa_bool
3225   \bool_while_do:nn {
3226     \str_if_eq_p:ee X {
3227       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3228     }
3229   }{
3230     \int_incr:N \l_tmpa_int
3231   }
3232
3233   \peek_charcode:NTF [ {
3234     % notation/text component
3235     \__stex_terms_custom_component:w
3236   } {
3237     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3238       % all arguments read => finish
3239       \__stex_terms_custom_final:

```

```

3240 } {
3241   % arguments missing
3242   \peek_charcode_remove:NTF * {
3243     % invisible, specific argument position or both
3244     \peek_charcode:NTF [ {
3245       % visible specific argument position
3246       \__stex_terms_custom_arg:wn
3247     } {
3248       % invisible
3249       \peek_charcode_remove:NTF * {
3250         % invisible specific argument position
3251         \__stex_terms_custom_arg_inv:wn
3252       } {
3253         % invisible next argument
3254         \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3255       }
3256     }
3257   } {
3258     % next normal argument
3259     \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3260   }
3261 }
3262 }
3263 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

3264 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3265   \bool_set_true:N \l_tmpa_bool
3266   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
3267 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

3268 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
3269   \str_set:Nx \l_tmpb_str {
3270     \str_item:Nn \l_tmpa_str { #1 }
3271   }
3272   \str_case:VnTF \l_tmpb_str {
3273     { X } {
3274       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3275     }
3276     { i } { \__stex_terms_custom_set_X:n { #1 } }
3277     { b } { \__stex_terms_custom_set_X:n { #1 } }
3278     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3279     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
3280   }{}{
3281     \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
3282   }
3283
3284   \bool_if:nTF \l_tmpa_bool {
3285     \tl_put_right:Nx \l_tmpa_tl {
3286       \stex_annotate_invisible:n {

```

```

3287         \stex_term_arg:nn { \int_eval:n { #1 } }
3288         \exp_not:n { { #2 } }
3289     }
3290 }
3291 } {
3292     \tl_put_right:Nx \l_tmpa_tl {
3293         \stex_term_arg:nn { \int_eval:n { #1 } }
3294         \exp_not:n { { #2 } }
3295     }
3296 }
3297
3298 \__stex_terms_custom_loop:
3299 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

3300 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
3301     \str_set:Nx \l_tmpa_str {
3302         \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3303         X
3304         \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3305     }
3306 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

3307 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
3308     \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3309     \__stex_terms_custom_loop:
3310 }

```

(End definition for __stex_terms_custom_component:.)

__stex_terms_custom_final:

```

3311 \cs_new_protected:Nn \__stex_terms_custom_final: {
3312     \int_compare:nNnTF \l_tmpb_int = 0 {
3313         \exp_args:Nnno \stex_term_oms:nnn
3314     }{
3315         \str_if_in:NnTF \l_tmpa_str {b} {
3316             \exp_args:Nnno \stex_term_ombind:nnn
3317         } {
3318             \exp_args:Nnno \stex_term_oma:nnn
3319         }
3320     }
3321     { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3322 }

```

(End definition for __stex_terms_custom_final:.)

\symref

\symname

```

3323 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3324
3325 \keys_define:nn { stex / symname } {

```



```

3326 pre .tl_set_x:N = \l__stex_terms_pre_tl ,
3327 post .tl_set_x:N = \l__stex_terms_post_tl ,
3328 root .tl_set_x:N = \l__stex_terms_root_tl
3329 }
3330
3331 \cs_new_protected:Nn \stex_symname_args:n {
3332 \tl_clear:N \l__stex_terms_post_tl
3333 \tl_clear:N \l__stex_terms_pre_tl
3334 \tl_clear:N \l__stex_terms_root_str
3335 \keys_set:nn { stex / symname } { #1 }
3336 }
3337
3338 \NewDocumentCommand \symref { m m }{
3339 \let\compemph_uri_prev:\compemph@uri
3340 \let\compemph@uri\symrefemph@uri
3341 \STEXsymbol{#1}!\{ #2 }
3342 \let\compemph@uri\compemph_uri_prev:
3343 }
3344
3345 \NewDocumentCommand \synonym { 0{} m m }{
3346 \stex_symname_args:n { #1 }
3347 \let\compemph_uri_prev:\compemph@uri
3348 \let\compemph@uri\symrefemph@uri
3349 % TODO
3350 \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3351 \let\compemph@uri\compemph_uri_prev:
3352 }
3353
3354 \NewDocumentCommand \symname { 0{} m }{
3355 \stex_symname_args:n { #1 }
3356 \stex_get_symbol:n { #2 }
3357 \str_set:Nx \l_tmpa_str {
3358 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3359 }
3360 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3361
3362 \let\compemph_uri_prev:\compemph@uri
3363 \let\compemph@uri\symrefemph@uri
3364 \exp_args:NNx \use:nn
3365 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3366 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3367 } }
3368 \let\compemph@uri\compemph_uri_prev:
3369 }
3370
3371 \NewDocumentCommand \Symname { 0{} m }{
3372 \stex_symname_args:n { #1 }
3373 \stex_get_symbol:n { #2 }
3374 \str_set:Nx \l_tmpa_str {
3375 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3376 }
3377 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3378 \let\compemph_uri_prev:\compemph@uri
3379 \let\compemph@uri\symrefemph@uri

```

```

3380 \exp_args:Nnx \use:nn
3381 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3382   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3383   \l__stex_terms_post_tl
3384 } }
3385 \let\compemph@uri\compemph_uri_prev:
3386 }

```

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

31.3 Notation Components

```

3387 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3388
3389 \str_new:N \l_stex_current_symbol_str
3390 \cs_new_protected:Nn \stex_highlight_term:nn {
3391   \exp_args:Nnx
3392   \use:nn {
3393     \str_set:Nx \l_stex_current_symbol_str { #1 }
3394     #2
3395   } {
3396     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3397     { \l_stex_current_symbol_str }
3398   }
3399 }
3400
3401 \cs_new_protected:Nn \stex_unhighlight_term:n {
3402   % \latexml_if:TF {
3403   %   #1
3404   % } {
3405   %   \rustex_if:TF {
3406   %     #1
3407   %   } {
3408   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3409   %   }
3410   % }
3411 }

```

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

```

\comp
\compemph@uri 3412 \cs_new_protected:Npn \comp #1 {
\compemph 3413   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3414     \rustex_if:TF {
\defemph@uri 3415       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3416     }{
\symrefemph@uri 3417       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3418     }
3419   }
3420 }
3421
3422 \cs_new_protected:Npn \compemph@uri #1 #2 {

```

```

3423     \compemph{ #1 }
3424 }
3425
3426
3427 \cs_new_protected:Npn \compemph #1 {
3428     #1
3429 }
3430
3431 \cs_new_protected:Npn \defemph@uri #1 #2 {
3432     \defemph{#1}
3433 }
3434
3435 \cs_new_protected:Npn \defemph #1 {
3436     \textbf{#1}
3437 }
3438
3439 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3440     \symrefemph{#1}
3441 }
3442
3443 \cs_new_protected:Npn \symrefemph #1 {
3444     \textbf{#1}
3445 }

```

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

`\ellipses`

```

3446 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3447 \bool_new:N \l_stex_inarray_bool
3448 \bool_set_false:N \l_stex_inarray_bool
3449 \NewDocumentCommand \parray { m m } {
3450     \begingroup
3451     \bool_set_true:N \l_stex_inarray_bool
3452     \begin{array}{#1}
3453         #2
3454     \end{array}
3455 \endgroup
3456 }
3457
3458 \NewDocumentCommand \prmatrix { m } {
3459     \begingroup
3460     \bool_set_true:N \l_stex_inarray_bool
3461     \begin{matrix}
3462         #1
3463     \end{matrix}
3464 \endgroup
3465 }
3466
3467 \def \maybepline {
3468     \bool_if:NT \l_stex_inarray_bool {\hline}
3469 }

```

```

3470
3471 \def \parrayline #1 #2 {
3472   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3473 }
3474
3475 \def \pmrow #1 { \parrayline{}{ #1 } }
3476
3477 \def \parraylineh #1 #2 {
3478   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3479 }
3480
3481 \def \parraycell #1 {
3482   #1 \bool_if:NT \l_stex_inarray_bool {&}
3483 }

```

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

31.4 Variables

```

3484 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3485 \cs_new_protected:Nn \stex_invoke_variable:n {
3486   \if_mode_math:
3487     \exp_after:wN \__stex_variables_invoke_math:n
3488   \else:
3489     \exp_after:wN \__stex_variables_invoke_text:n
3490   \fi: {#1}
3491 }
3492
3493 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3494   %TODO
3495 }
3496
3497
3498 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3499   \peek_charcode_remove:NTF ! {
3500     \peek_charcode_remove:NTF ! {
3501       \peek_charcode:NTF [ {
3502         \__stex_variables_invoke_op_custom:nw
3503       }{
3504         % TODO throw error
3505       }
3506     }{
3507       \__stex_variables_invoke_op:n { #1 }
3508     }
3509   }{
3510     \peek_charcode_remove:NTF * {
3511       \__stex_variables_invoke_text:n { #1 }
3512     }{
3513       \__stex_variables_invoke_math_ii:n { #1 }
3514     }
3515   }
3516 }

```

```

3517
3518 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3519   \cs_if_exist:cTF {
3520     stex_var_op_notation_ #1 _cs
3521   }{
3522     \use:c{stex_var_op_notation_ #1 _cs }
3523   }{
3524     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3525   }
3526 }
3527
3528 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3529   \cs_if_exist:cTF {
3530     stex_var_notation_#1_cs
3531   }{
3532     \str_set:Nn \l_stex_current_symbol_str { #1 }
3533     \use:c{stex_var_notation_#1_cs}
3534   }{
3535     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3536   }
3537 }

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

3538 \</package>

```

Chapter 32

STEX -Structural Features Implementation

```
3539 <*package>
3540
3541 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3542
3543 <@@=stex_features>
3544
3545   Warnings and error messages
3546 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3547   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3548 }
3549 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3550   Symbol~#1~not~assigned~in~interpretmodule~#2
3551 }
```

32.1 Imports with modification

```
3551 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3552   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3553     \__stex_features_get_symbol_from_cs:n { #1 }
3554   }{
3555     % argument is a string
3556     % is it a command name?
3557     \cs_if_exist:cTF { #1 }{
3558       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3559       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3560       \str_if_empty:NNTF \l_tmpa_str {
3561         \exp_args:Nx \cs_if_eq:NNTF {
3562           \tl_head:N \l_tmpa_tl
3563         } \stex_invoke_symbol:n {
3564           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3565         }{
3566           \__stex_features_get_symbol_from_string:n { #1 }
3567         }
3568       }
3569     }
3570   }
```

```

3567     }
3568   } {
3569     \__stex_features_get_symbol_from_string:n { #1 }
3570   }
3571   }{
3572     % argument is not a command name
3573     \__stex_features_get_symbol_from_string:n { #1 }
3574     % \l_stex_all_symbols_seq
3575   }
3576 }
3577 }
3578
3579 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3580   \str_set:Nn \l_tmpa_str { #1 }
3581   \bool_set_false:N \l_tmpa_bool
3582   \bool_if:NF \l_tmpa_bool {
3583     \tl_set:Nn \l_tmpa_tl {
3584       \msg_set:nnn{stex}{error/unknownsymbol}{
3585         No~symbol~#1~found!
3586       }
3587       \msg_error:nn{stex}{error/unknownsymbol}
3588     }
3589     \str_set:Nn \l_tmpa_str { #1 }
3590     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3591     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3592       \str_set:Nn \l_tmpb_str { ##1 }
3593       \str_if_eq:eeT { \l_tmpa_str } {
3594         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3595       } {
3596         \seq_map_break:n {
3597           \tl_set:Nn \l_tmpa_tl {
3598             \str_set:Nn \l_stex_get_symbol_uri_str {
3599               ##1
3600             }
3601             \__stex_features_get_symbol_check:
3602           }
3603         }
3604       }
3605     }
3606     \l_tmpa_tl
3607   }
3608 }
3609
3610 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3611   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3612   { \tl_tail:N \l_tmpa_tl }
3613   \tl_if_single:NTF \l_tmpa_tl {
3614     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3615       \exp_after:wN \str_set:Nn \exp_after:wN
3616       \l_stex_get_symbol_uri_str \l_tmpa_tl
3617       \__stex_features_get_symbol_check:
3618     }{
3619       % TODO
3620       % tail is not a single group

```

```

3621     }
3622   }{
3623     % TODO
3624     % tail is not a single group
3625   }
3626 }
3627
3628 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3629   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3630   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3631     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3632     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3633     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3634       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3635         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3636       }
3637     }
3638   }{
3639     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3640       \l_stex_current_copymodule_name_str~(inexplicably)
3641     }
3642   }
3643 }
3644
3645 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3646   \stex_import_module_uri:nn { #1 } { #2 }
3647   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3648   \stex_import_require_module:nnnn
3649     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3650     { \l_stex_import_path_str } { \l_stex_import_name_str }
3651   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3652   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3653   \seq_clear:N \l__stex_features_copymodule_fields_seq
3654   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3655     \seq_map_inline:cn {c_stex_module_###1_constants}{
3656       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3657         ###1 ? #####1
3658       }
3659     }
3660   }
3661   \seq_clear:N \l_tmpa_seq
3662   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3663     name      = \l_stex_current_copymodule_name_str ,
3664     module    = \l_stex_current_module_str ,
3665     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3666     includes  = \l_tmpa_seq ,
3667     fields    = \l_tmpa_seq
3668   }
3669   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3670     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3671   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3672     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3673   \stex_if_smsmode:F {
3674     \begin{stex_annotate_env} {#4} {

```



```

3675     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3676   }
3677   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3678 }
3679 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3680 \bool_set_false:N \stex_html_do_output_bool
3681 }
3682 \cs_new_protected:Nn \stex_copymodule_end:n {
3683   \def \l_tmpa_cs ##1 ##2 {#1}
3684   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3685   \tl_clear:N \l_tmpa_tl
3686   \tl_clear:N \l_tmpb_tl
3687   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3688   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3689     \seq_map_inline:cn {c_stex_module_##1_constants}{
3690       \tl_clear:N \l_tmpc_tl
3691       \l_tmpa_cs{##1}{####1}
3692       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3693         \tl_put_right:Nx \l_tmpa_tl {
3694           \prop_set_from_keyval:cn {
3695             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3696           }{
3697             \exp_after:wN \prop_to_keyval:N \csname
3698               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3699             \endcsname
3700           }
3701           \seq_clear:c {
3702             l_stex_symdecl_
3703             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3704             _notations
3705           }
3706         }
3707         \tl_put_right:Nx \l_tmpc_tl {
3708           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_cop
3709           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3710         }
3711         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3712         \str_if_exist:cT {\l_stex_features_copymodule_##1?####1_macroname_str} {
3713           \tl_put_right:Nx \l_tmpc_tl {
3714             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3715           }
3716           \tl_put_right:Nx \l_tmpa_tl {
3717             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3718             \stex_invoke_symbol:n {
3719               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3720             }
3721           }
3722         }
3723       }
3724     }{
3725       \tl_put_right:Nx \l_tmpc_tl {
3726         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3727       }
3728       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3729 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3730 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3731 \tl_put_right:Nx \l_tmpa_tl {
3732   \prop_set_from_keyval:cn {
3733     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3734   }{
3735     \prop_to_keyval:N \l_tmpa_prop
3736   }
3737   \seq_clear:c {
3738     l_stex_symdecl_
3739     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3740     _notations
3741   }
3742 }
3743 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3744 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3745   \tl_put_right:Nx \l_tmpc_tl {
3746     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3747   }
3748   \tl_put_right:Nx \l_tmpa_tl {
3749     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3750       \stex_invoke_symbol:n {
3751         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3752       }
3753     }
3754   }
3755 }
3756 }
3757 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3758   \tl_put_right:Nx \l_tmpc_tl {
3759     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3760   }
3761 }
3762 \tl_put_right:Nx \l_tmpb_tl {
3763   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3764 }
3765 }
3766 }
3767 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3768 \tl_put_left:Nx \l_tmpa_tl {
3769   \prop_set_from_keyval:cn {
3770     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3771   }{
3772     \prop_to_keyval:N \l_stex_current_copymodule_prop
3773   }
3774 }
3775 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3776 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3777 \exp_args:Nx \stex_do_up_to_module:n {
3778   \exp_args:No \exp_not:n \l_tmpa_tl
3779 }
3780 \l_tmpb_tl
3781 \stex_if_smsmode:F {
3782   \end{stex_annotate_env}

```

```

3783 }
3784 }
3785
3786 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3787   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3788   \stex_deactivate_macro:Nn \symdecl {module~environments}
3789   \stex_deactivate_macro:Nn \symdef {module~environments}
3790   \stex_deactivate_macro:Nn \notation {module~environments}
3791   \stex_reactivate_macro:N \assign
3792   \stex_reactivate_macro:N \renamedec1
3793   \stex_reactivate_macro:N \donotcopy
3794   \stex_smsmode_do:
3795 }{
3796   \stex_copymodule_end:n {}
3797 }
3798
3799 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3800   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3801   \stex_deactivate_macro:Nn \symdecl {module~environments}
3802   \stex_deactivate_macro:Nn \symdef {module~environments}
3803   \stex_deactivate_macro:Nn \notation {module~environments}
3804   \stex_reactivate_macro:N \assign
3805   \stex_reactivate_macro:N \renamedec1
3806   \stex_reactivate_macro:N \donotcopy
3807   \stex_smsmode_do:
3808 }{
3809   \stex_copymodule_end:n {
3810     \tl_if_exist:cF {
3811       l__stex_features_copymodule_##1?##2_def_tl
3812     }{
3813       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3814         ##1?##2
3815       }{\l_stex_current_copymodule_name_str}
3816     }
3817   }
3818 }
3819
3820 \NewDocumentCommand \donotcopy { 0{} m}{
3821   \stex_import_module_uri:nn { #1 } { #2 }
3822   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3823   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3824     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3825     \seq_map_inline:cn {c_stex_module_##1_constants}{
3826       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3827       \bool_lazy_any_p:nT {
3828         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3829         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3830         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3831       }{
3832         % TODO throw error
3833       }
3834     }
3835   }
3836 }

```

```

3837 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3838 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3839 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3840 }
3841
3842 \NewDocumentCommand \assign { m m }{
3843   \stex_get_symbol_in_copymodule:n {#1}
3844   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3845   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3846 }
3847
3848 \keys_define:nn { stex / renamedec1 } {
3849   name .str_set_x:N = \l_stex_renamedec1_name_str
3850 }
3851 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3852   \str_clear:N \l_stex_renamedec1_name_str
3853
3854   \keys_set:nn { stex / renamedec1 } { #1 }
3855 }
3856
3857 \NewDocumentCommand \renamedec1 { 0{} m m }{
3858   \__stex_features_renamedec1_args:n { #1 }
3859   \stex_get_symbol_in_copymodule:n {#2}
3860   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3861   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3862   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3863     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3864       \l_stex_get_symbol_uri_str
3865     } }
3866   } {
3867     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3868     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3869     \prop_set_eq:cc {l_stex_symdecl_
3870       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3871       _prop
3872     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3873     \seq_set_eq:cc {l_stex_symdecl_
3874       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3875       _notations
3876     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3877     \prop_put:cnx {l_stex_symdecl_
3878       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3879       _prop
3880     }{ name }{ \l_stex_renamedec1_name_str }
3881     \prop_put:cnx {l_stex_symdecl_
3882       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3883       _prop
3884     }{ module }{ \l_stex_current_module_str }
3885     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3886       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3887     }
3888     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3889       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3890     } }

```

```

3891 }
3892 }
3893 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3894 % \_stex_notation_args:n { #1 }
3895 % \tl_clear:N \l_stex_symdecl_definiens_tl
3896 % \stex_get_symbol_in_copymodule:n { #2 }
3897 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3898 % % todo
3899 %}
3900 \stex_deactivate_macro:Nn \assign {copymodules}
3901 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3902 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3903
3904
3905 \seq_new:N \l_stex_implicit_morphisms_seq
3906 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3907 \stex_import_module_uri:nn { #1 } { #2 }
3908 \stex_debug:nn{implicits}{
3909 Implicit~morphism:~
3910 \l_stex_module_ns_str ? \l__stex_features_name_str
3911 }
3912 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3913 \l_stex_module_ns_str ? \l__stex_features_name_str
3914 }{
3915 \msg_error:nnn{stex}{error/conflictingmodules}{
3916 \l_stex_module_ns_str ? \l__stex_features_name_str
3917 }
3918 }
3919
3920 % TODO
3921
3922
3923
3924 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3925 \l_stex_module_ns_str ? \l__stex_features_name_str
3926 }
3927 }
3928

```

32.2 The feature environment

structural@feature

```

3929
3930 \NewDocumentEnvironment{structural@feature}{ m m m }{
3931 \stex_if_in_module:F {
3932 \msg_set:nnn{stex}{error/nomodule}{
3933 Structural~Feature~has~to~occur~in~a~module:\\
3934 Feature~#2~of~type~#1\\
3935 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3936 }
3937 \msg_error:nn{stex}{error/nomodule}
3938 }
3939

```

```

3940 \str_set:Nx \l_stex_module_name_str {
3941   \prop_item:Nn \l_stex_current_module_prop
3942     { name } / #2 - feature
3943 }
3944
3945 \str_set:Nx \l_stex_module_ns_str {
3946   \prop_item:Nn \l_stex_current_module_prop
3947     { ns }
3948 }
3949
3950
3951 \str_clear:N \l_tmpa_str
3952 \seq_clear:N \l_tmpa_seq
3953 \tl_clear:N \l_tmpa_tl
3954 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3955   origname = #2,
3956   name     = \l_stex_module_name_str ,
3957   ns       = \l_stex_module_ns_str ,
3958   imports  = \exp_not:o { \l_tmpa_seq } ,
3959   constants = \exp_not:o { \l_tmpa_seq } ,
3960   content  = \exp_not:o { \l_tmpa_tl } ,
3961   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3962   lang     = \l_stex_module_lang_str ,
3963   sig      = \l_tmpa_str ,
3964   meta     = \l_tmpa_str ,
3965   feature  = #1 ,
3966 }
3967
3968 \stex_if_smsmode:F {
3969   \begin{stex_annotate_env}{ feature:#1 }{}
3970   \stex_annotate_invisible:nnn{header}{}{ #3 }
3971 }
3972 }{
3973   \str_set:Nx \l_tmpa_str {
3974     c_stex_feature_
3975     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3976     \prop_item:Nn \l_stex_current_module_prop { name }
3977     _prop
3978   }
3979   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3980   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3981   \stex_if_smsmode:F {
3982     \end{stex_annotate_env}
3983   }
3984 }
3985

```

32.3 Features

structure

```

3986
3987 \prop_new:N \l_stex_all_structures_prop
3988

```

```

3989 \keys_define:nn { stex / features / structure } {
3990   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3991 }
3992
3993 \cs_new_protected:Nn \__stex_features_structure_args:n {
3994   \str_clear:N \l__stex_features_structure_name_str
3995   \keys_set:nn { stex / features / structure } { #1 }
3996 }
3997
3998 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3999 %   \__stex_features_structure_args:n { ##1 }
4000 %   \str_if_empty:NT \l__stex_features_structure_name_str {
4001 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
4002 %   }
4003 % } {
4004 %
4005 %}
4006
4007 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
4008   \__stex_features_structure_args:n { #1 }
4009   \str_if_empty:NT \l__stex_features_structure_name_str {
4010     \str_set:Nx \l__stex_features_structure_name_str { #2 }
4011   }
4012   \exp_args:Nnnx
4013   \begin{structural@feature}{ structure }
4014     { \l__stex_features_structure_name_str }{ }
4015     \seq_clear:N \l_tmpa_seq
4016     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
4017     \stex_smsmode_do:
4018   }{
4019     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
4020     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
4021     \str_set:Nx \l_tmpa_str {
4022       \prop_item:Nn \l_stex_current_module_prop { ns } ?
4023       \prop_item:Nn \l_stex_current_module_prop { name }
4024     }
4025     \seq_map_inline:Nn \l_tmpa_seq {
4026       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
4027     }
4028     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
4029     \exp_args:Nnx
4030     \AddToHookNext { env / mathstructure / after }{
4031       \symdecl{ #2 }[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
4032         \stex_term_math_oms:nnnn { \l_tmpa_str }{ }{0}{ }
4033       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]
4034       \STEXexport {
4035         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4036         { \prop_item:Nn \l_stex_current_module_prop { origname } }
4037         { \l_tmpa_str }
4038         \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
4039         { #2 }{ \l_tmpa_str }
4040       %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4041       %     \prop_item:Nn \l_stex_current_module_prop { origname },
4042       %     \l_tmpa_str

```

```

4043 %     }
4044 %     \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
4045 %         #2,\l_tmpa_str
4046 %     }
4047 %     \tl_set:cx { #2 } {
4048 %         \stex_invoke_structure:n { \l_tmpa_str }
4049 %     }
4050 % }
4051
4052 \end{structural@feature}
4053 % \g_stex_last_feature_prop
4054 }

```

\instantiate

```

4055 \seq_new:N \l__stex_features_structure_field_seq
4056 \str_new:N \l__stex_features_structure_field_str
4057 \str_new:N \l__stex_features_structure_def_tl
4058 \prop_new:N \l__stex_features_structure_prop
4059 \NewDocumentCommand \instantiate { m O{} m }{
4060     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
4061     \prop_set_eq:Nc \l__stex_features_structure_prop {
4062         c_stex_feature_\l_tmpa_str _prop
4063     }
4064     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
4065     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
4066         \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
4067         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4068             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
4069             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
4070                 {!} \l_tmpa_tl
4071             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4072                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
4073                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4074                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4075             }{
4076                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
4077                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
4078                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
4079                     \l_tmpa_tl
4080                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
4081                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
4082                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
4083                 }{
4084                     \tl_clear:N \l_tmpb_tl
4085                 }
4086             }
4087         }{
4088             \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
4089             \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
4090                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
4091                 \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
4092                 \tl_clear:N \l_tmpa_tl
4093             }{
4094                 % TODO throw error

```



```

4095     }
4096   }
4097   % \l_tmpa_str: name
4098   % \l_tmpa_tl: definiens
4099   % \l_tmpb_tl: notation
4100   \tl_if_empty:NT \l__stex_features_structure_field_str {
4101     % TODO throw error
4102   }
4103   \str_clear:N \l_tmpb_str
4104
4105   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4106   \seq_map_inline:Nn \l_tmpa_seq {
4107     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
4108     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
4109     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
4110       \seq_map_break:n {
4111         \str_set:Nn \l_tmpb_str { ####1 }
4112       }
4113     }
4114   }
4115   \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
4116   \l_tmpb_str
4117
4118   \tl_if_empty:NTF \l_tmpb_tl {
4119     \tl_if_empty:NF \l_tmpa_tl {
4120       \exp_args:Nx \use:n {
4121         \symdecl{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4122       }
4123     }
4124   }{
4125     \tl_if_empty:NTF \l_tmpa_tl {
4126       \exp_args:Nx \use:n {
4127         \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str]\exp_after:wN\
4128       }
4129     }{
4130       \exp_args:Nx \use:n {
4131         \symdef{#3/\l__stex_features_structure_field_str}[args=\l_tmpb_str,def={\exp_args:
4132         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
4133       }
4134     }
4135   }
4136 }
4137 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
4138 % \prop_item:Nn \l_stex_current_module_prop {name} ?
4139 % #3/\l__stex_features_structure_field_str
4140 % \par
4141 % \expandafter\present\csname
4142 %   l_stex_symdecl_
4143 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
4144 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
4145 %   #3/\l__stex_features_structure_field_str
4146 %   _prop
4147 % \endcsname
4148 }

```

```

4149
4150 \tl_clear:N \l__stex_features_structure_def_tl
4151
4152 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4153 \seq_map_inline:Nn \l_tmpa_seq {
4154   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4155   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4156   \exp_args:Nx \use:n {
4157     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
4158
4159       }
4160     }
4161
4162   \prop_if_exist:cF {
4163     l_stex_symdecl_
4164     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4165     \prop_item:Nn \l_stex_current_module_prop {name} ?
4166     #3/\l_tmpa_str
4167     _prop
4168   }{
4169     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
4170     \l_tmpb_str
4171     \exp_args:Nx \use:n {
4172       \symdecl{#3/\l_tmpa_str}[args=\l_tmpb_str]
4173     }
4174   }
4175 }
4176
4177 \symdecl*{#3}[type={\STEXsymbol{module-type}}{
4178   \stex_term_math_oms:nnnn {
4179     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4180     \prop_item:Nn \l__stex_features_structure_prop {name}
4181     }{0}{0}}
4182 }]}
4183
4184 % TODO: -> sms file
4185
4186 \tl_set:cx{ #3 }{
4187   \stex_invoke_structure:nnn {
4188     \prop_item:Nn \l_stex_current_module_prop {ns} ?
4189     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
4190   } {
4191     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
4192     \prop_item:Nn \l__stex_features_structure_prop {name}
4193   }
4194 }
4195 \stex_smsmode_do:
4196 }

```

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

\stex_invoke_structure:nnn

```

4197 % #1: URI of the instance
4198 % #2: URI of the instantiated module

```

```

4199 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4200   \tl_if_empty:nTF{ #3 }{
4201     \prop_set_eq:Nc \l__stex_features_structure_prop {
4202       c_stex_feature_ #2 _prop
4203     }
4204     \tl_clear:N \l_tmpa_tl
4205     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
4206     \seq_map_inline:Nn \l_tmpa_seq {
4207       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4208       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4209       \cs_if_exist:cT {
4210         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4211       }{
4212         \tl_if_empty:NF \l_tmpa_tl {
4213           \tl_put_right:Nn \l_tmpa_tl {,}
4214         }
4215         \tl_put_right:Nx \l_tmpa_tl {
4216           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4217         }
4218       }
4219     }
4220     \exp_args:No \mathstrut \l_tmpa_tl
4221   }{
4222     \stex_invoke_symbol:n{#1/#3}
4223   }
4224 }

```

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

```

4225 </package>

```

Chapter 33

STEX -Statements Implementation

```
4226 <*package>
4227
4228 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4229
4230 <@@=stex_statements>
    Warnings and error messages
4231
\titleemph
4232 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
4233 \keys_define:nn {stex / definiendum }{
4234   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4235   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4236   root     .str_set_x:N   = \l__stex_statements_definiendum_root_str,
4237   gfa      .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
4238 }
4239 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4240   \str_clear:N \l__stex_statements_definiendum_root_str
4241   \tl_clear:N \l__stex_statements_definiendum_post_tl
4242   \str_clear:N \l__stex_statements_definiendum_gfa_str
4243   \keys_set:nn { stex / definiendum }{ #1 }
4244 }
4245 \NewDocumentCommand \definiendum { O{} m m } {
4246   \__stex_statements_definiendum_args:n { #1 }
4247   \stex_get_symbol:n { #2 }
4248   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4249   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4250     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4251     \tl_set:Nn \l_tmpa_tl { #3 }
4252   } {
4253     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4254     \tl_set:Nn \l_tmpa_tl {
4255       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4256     }
4257   }
4258 } {
4259   \tl_set:Nn \l_tmpa_tl { #3 }
4260 }
4261
4262 % TODO root
4263 \rustex_if:TF {
4264   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4265 } {
4266   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4267 }
4268 }
4269 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4270
4271 \NewDocumentCommand \definame { 0{ } m } {
4272   \__stex_statements_definiendum_args:n { #1 }
4273   % TODO: root
4274   \stex_get_symbol:n { #2 }
4275   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4276   \str_set:Nx \l_tmpa_str {
4277     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4278   }
4279   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4280   \rustex_if:TF {
4281     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4282       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4283     }
4284   } {
4285     \defemph@uri {
4286       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4287     } { \l_stex_get_symbol_uri_str }
4288   }
4289 }
4290 \stex_deactivate_macro:Nn \definame {definition~environments}
4291
4292 \NewDocumentCommand \Definame { 0{ } m } {
4293   \__stex_statements_definiendum_args:n { #1 }
4294   \stex_get_symbol:n { #2 }
4295   \str_set:Nx \l_tmpa_str {
4296     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4297   }
4298   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4299   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4300   \rustex_if:TF {

```

```

4301 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4302 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4303 }
4304 } {
4305 \defemph@uri {
4306 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4307 } { \l_stex_get_symbol_uri_str }
4308 }
4309 }
4310 \stex_deactivate_macro:Nn \Definame {definition~environments}
4311
4312 \NewDocumentCommand \premise { m }{
4313 \stex_annotate:nnn{ premise }{}{ #1 }
4314 }
4315 \NewDocumentCommand \conclusion { m }{
4316 \stex_annotate:nnn{ conclusion }{}{ #1 }
4317 }
4318 \NewDocumentCommand \definiens { m }{
4319 \stex_annotate:nnn{ definiens }{}{ #1 }
4320 }
4321
4322 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4323 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4324 \stex_deactivate_macro:Nn \definiens {definition~environments}
4325

```

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

sdefinition

```

4326
4327 \keys_define:nn {stex / sdefinition }{
4328 type .str_set_x:N = \sdefinitiontype,
4329 id .str_set_x:N = \sdefinitionid,
4330 name .str_set_x:N = \sdefinitionname,
4331 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4332 title .tl_set:N = \sdefinitiontitle
4333 }
4334 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4335 \str_clear:N \sdefinitiontype
4336 \str_clear:N \sdefinitionid
4337 \str_clear:N \sdefinitionname
4338 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4339 \tl_clear:N \sdefinitiontitle
4340 \keys_set:nn { stex / sdefinition }{}{ #1 }
4341 }
4342
4343 \NewDocumentEnvironment{sdefinition}{0{}}{
4344 \__stex_statements_sdefinition_args:n{ #1 }
4345 \stex_reactivate_macro:N \definiendum
4346 \stex_reactivate_macro:N \definame
4347 \stex_reactivate_macro:N \Definame
4348 \stex_reactivate_macro:N \premise
4349 \stex_reactivate_macro:N \definiens
4350 \stex_if_smsmode:F{

```

```

4351 \seq_clear:N \l_tmpa_seq
4352 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4353   \tl_if_empty:nF{ ##1 }{
4354     \stex_get_symbol:n { ##1 }
4355     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4356       \l_stex_get_symbol_uri_str
4357     }
4358   }
4359 }
4360 \exp_args:Nnnx
4361 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4362 \str_if_empty:NF \sdefinitiontype {
4363   \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4364 }
4365 \clist_set:No \l_tmpa_clist \sdefinitiontype
4366 \tl_clear:N \l_tmpa_tl
4367 \clist_map_inline:Nn \l_tmpa_clist {
4368   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4369     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4370   }
4371 }
4372 \tl_if_empty:NTF \l_tmpa_tl {
4373   \__stex_statements_sdefinition_start:
4374 }{
4375   \l_tmpa_tl
4376 }
4377 }
4378 \stex_ref_new_doc_target:n \sdefinitionid
4379 \stex_smsmode_do:
4380 }{
4381   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4382   \stex_if_smsmode:F {
4383     \clist_set:No \l_tmpa_clist \sdefinitiontype
4384     \tl_clear:N \l_tmpa_tl
4385     \clist_map_inline:Nn \l_tmpa_clist {
4386       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4387         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4388       }
4389     }
4390     \tl_if_empty:NTF \l_tmpa_tl {
4391       \__stex_statements_sdefinition_end:
4392     }{
4393       \l_tmpa_tl
4394     }
4395     \end{stex_annotate_env}
4396   }
4397 }

```

\stexpatchdefinition

```

4398 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4399   \par\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
4400     ~(\sdefinitiontitle)
4401   }~}
4402 }

```

```

4403 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4404
4405 \newcommand\stexpatchdefinition[3] [] {
4406   \str_set:Nx \l_tmpa_str{ #1 }
4407   \str_if_empty:NTF \l_tmpa_str {
4408     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4409     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4410   }{
4411     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4412     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4413   }
4414 }

```

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

\inlinedef inline:

```

4415 \keys_define:nn {stex / inlinedef }{
4416   type      .str_set_x:N = \sdefinitiontype,
4417   id        .str_set_x:N = \sdefinitionid,
4418   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4419   name      .str_set_x:N = \sdefinitionname
4420 }
4421 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4422   \str_clear:N \sdefinitiontype
4423   \str_clear:N \sdefinitionid
4424   \str_clear:N \sdefinitionname
4425   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4426   \keys_set:nn { stex / inlinedef }{ #1 }
4427 }
4428 \NewDocumentCommand \inlinedef { 0{} m } {
4429   \beginngroup
4430   \__stex_statements_inlinedef_args:n{ #1 }
4431   \stex_reactivate_macro:N \definiendum
4432   \stex_reactivate_macro:N \definame
4433   \stex_reactivate_macro:N \Definame
4434   \stex_reactivate_macro:N \premise
4435   \stex_reactivate_macro:N \definiens
4436   \stex_ref_new_doc_target:n \sdefinitionid
4437   \stex_if_smsmode:TF{
4438     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4439   }{
4440     \seq_clear:N \l_tmpa_seq
4441     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4442       \tl_if_empty:nF{ ##1 }{
4443         \stex_get_symbol:n { ##1 }
4444         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4445           \l_stex_get_symbol_uri_str
4446         }
4447       }
4448     }
4449     \exp_args:Nnx
4450     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4451       \str_if_empty:NF \sdefinitiontype {
4452         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{

```



```

4453     }
4454     #2
4455     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4456   }
4457 }
4458 \endgroup
4459 \stex_smsmode_do:
4460 }

```

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

33.2 Assertions

`sassertion`

```

4461
4462 \keys_define:nn {stex / sassertion }{
4463   type      .str_set_x:N = \sassertiontype,
4464   id        .str_set_x:N = \sassertionid,
4465   title     .tl_set:N    = \sassertiontitle ,
4466   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4467   name      .str_set_x:N = \sassertionname
4468 }
4469 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4470   \str_clear:N \sassertiontype
4471   \str_clear:N \sassertionid
4472   \str_clear:N \sassertionname
4473   \clist_clear:N \l__stex_statements_sassertion_for_clist
4474   \tl_clear:N \sassertiontitle
4475   \keys_set:nn { stex / sassertion }{ #1 }
4476 }
4477
4478 %\tl_new:N \g__stex_statements_aftergroup_tl
4479
4480 \NewDocumentEnvironment{sassertion}{0{}}{
4481   \__stex_statements_sassertion_args:n{ #1 }
4482   \stex_reactivate_macro:N \premise
4483   \stex_reactivate_macro:N \conclusion
4484   \stex_if_smsmode:F {
4485     \seq_clear:N \l_tmpa_seq
4486     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4487       \tl_if_empty:nF{ ##1 }{
4488         \stex_get_symbol:n { ##1 }
4489         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4490           \l_stex_get_symbol_uri_str
4491         }
4492       }
4493     }
4494     \exp_args:Nnnx
4495     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4496     \str_if_empty:NF \sassertiontype {
4497       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4498     }
4499     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

4500 \tl_clear:N \l_tmpa_tl
4501 \clist_map_inline:Nn \l_tmpa_clist {
4502   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4503     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4504   }
4505 }
4506 \tl_if_empty:NTF \l_tmpa_tl {
4507   \__stex_statements_sassertion_start:
4508 }{
4509   \l_tmpa_tl
4510 }
4511 }
4512 \str_if_empty:NTF \sassertionid {
4513   \str_if_empty:NF \sassertionname {
4514     \stex_ref_new_doc_target:n {}
4515   }
4516 } {
4517   \stex_ref_new_doc_target:n \sassertionid
4518 }
4519 \stex_smsmode_do:
4520 ){
4521   \str_if_empty:NF \sassertionname {
4522     \stex_symdecl_do:nn{ }\sassertionname}
4523   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4524 }
4525 \stex_if_smsmode:F {
4526   \clist_set:Nn \l_tmpa_clist \sassertiontype
4527   \tl_clear:N \l_tmpa_tl
4528   \clist_map_inline:Nn \l_tmpa_clist {
4529     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4530       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4531     }
4532   }
4533   \tl_if_empty:NTF \l_tmpa_tl {
4534     \__stex_statements_sassertion_end:
4535   }{
4536     \l_tmpa_tl
4537   }
4538   \end{stex_annotate_env}
4539 }
4540 }

```

\stexpatchassertion

```

4541
4542 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4543   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4544     (\sassertiontitle)
4545   }~}
4546 }
4547 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4548
4549 \newcommand\stexpatchassertion[3] [] {
4550   \str_set:Nx \l_tmpa_str{ #1 }
4551   \str_if_empty:NTF \l_tmpa_str {

```

```

4552     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4553     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4554   }{
4555     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4556     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4557   }
4558 }

```

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

\inlineass inline:

```

4559 \keys_define:nn {stex / inlineass }{
4560   type      .str_set_x:N = \sassertiontype,
4561   id        .str_set_x:N = \sassertionid,
4562   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4563   name      .str_set_x:N = \sassertionname
4564 }
4565 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4566   \str_clear:N \sassertiontype
4567   \str_clear:N \sassertionid
4568   \str_clear:N \sassertionname
4569   \clist_clear:N \l__stex_statements_sassertion_for_clist
4570   \keys_set:nn { stex / inlineass }{ #1 }
4571 }
4572 \NewDocumentCommand \inlineass { 0{} m } {
4573   \begingroup
4574   \stex_reactivate_macro:N \premise
4575   \stex_reactivate_macro:N \conclusion
4576   \__stex_statements_inlineass_args:n{ #1 }
4577   \str_if_empty:NTF \sassertionid {
4578     \str_if_empty:NF \sassertionname {
4579       \stex_ref_new_doc_target:n { }
4580     }
4581   } {
4582     \stex_ref_new_doc_target:n \sassertionid
4583   }
4584
4585   \stex_if_smsmode:TF{
4586     \str_if_empty:NF \sassertionname {
4587       \stex_symdecl_do:nn{}{\sassertionname}
4588       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4589     }
4590   }{
4591     \seq_clear:N \l_tmpa_seq
4592     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4593       \tl_if_empty:nF{ ##1 }{
4594         \stex_get_symbol:n { ##1 }
4595         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4596           \l_stex_get_symbol_uri_str
4597         }
4598       }
4599     }
4600     \exp_args:Nnx
4601     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{

```

```

4602     \str_if_empty:NF \sassertiontype {
4603       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4604     }
4605     #2
4606     \str_if_empty:NF \sassertionname {
4607       \stex_symdecl_do:nn{}{\sassertionname}
4608       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4609     }
4610   }
4611 }
4612 \endgroup
4613 \stex_smsmode_do:
4614 }

```

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

33.3 Examples

`sexample`

```

4615
4616 \keys_define:nn {stex / sexample }{
4617   type      .str_set_x:N = \exampletype,
4618   id        .str_set_x:N = \sexampleid,
4619   title     .tl_set:N    = \sexampletitle,
4620   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4621 }
4622 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4623   \str_clear:N \sexampletype
4624   \str_clear:N \sexampleid
4625   \tl_clear:N \sexampletitle
4626   \clist_clear:N \l__stex_statements_sexample_for_clist
4627   \keys_set:nn { stex / sexample }{ #1 }
4628 }
4629
4630 \NewDocumentEnvironment{sexample}{0{}}{
4631   \__stex_statements_sexample_args:n{ #1 }
4632   \stex_reactivate_macro:N \premise
4633   \stex_reactivate_macro:N \conclusion
4634   \stex_if_smsmode:F {
4635     \seq_clear:N \l_tmpa_seq
4636     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4637       \tl_if_empty:nF{ ##1 }{
4638         \stex_get_symbol:n { ##1 }
4639         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4640           \l_stex_get_symbol_uri_str
4641         }
4642       }
4643     }
4644     \exp_args:Nnnx
4645     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4646     \str_if_empty:NF \sexampletype {
4647       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4648     }

```

```

4649 \clist_set:No \l_tmpa_clist \sexamplotype
4650 \tl_clear:N \l_tmpa_tl
4651 \clist_map_inline:Nn \l_tmpa_clist {
4652   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4653     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4654   }
4655 }
4656 \tl_if_empty:NTF \l_tmpa_tl {
4657   \__stex_statements_sexample_start:
4658 }{
4659   \l_tmpa_tl
4660 }
4661 }
4662 \str_if_empty:NF \sexampleid {
4663   \stex_ref_new_doc_target:n \sexampleid
4664 }
4665 \stex_smsmode_do:
4666 }{
4667   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4668   \stex_if_smsmode:F {
4669     \clist_set:No \l_tmpa_clist \sexamplotype
4670     \tl_clear:N \l_tmpa_tl
4671     \clist_map_inline:Nn \l_tmpa_clist {
4672       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4673         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4674       }
4675     }
4676     \tl_if_empty:NTF \l_tmpa_tl {
4677       \__stex_statements_sexample_end:
4678     }{
4679       \l_tmpa_tl
4680     }
4681     \end{stex_annotate_env}
4682   }
4683 }

```

\stexpatchexample

```

4684
4685 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4686   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4687     (\sexamplename)
4688   }~}
4689 }
4690 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4691
4692 \newcommand\stexpatchexample[3]{} {
4693   \str_set:Nx \l_tmpa_str{ #1 }
4694   \str_if_empty:NTF \l_tmpa_str {
4695     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4696     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4697   }{
4698     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4699     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4700   }

```

4701 }

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

\inlineex inline:

```

4702 \keys_define:nn {stex / inlineex }{
4703   type      .str_set_x:N = \sexamplotype,
4704   id        .str_set_x:N = \sexampleid,
4705   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4706   name      .str_set_x:N = \sexamplename
4707 }
4708 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4709   \str_clear:N \sexamplotype
4710   \str_clear:N \sexampleid
4711   \str_clear:N \sexamplename
4712   \clist_clear:N \l__stex_statements_sexample_for_clist
4713   \keys_set:nn { stex / inlineex }{ #1 }
4714 }
4715 \NewDocumentCommand \inlineex { 0{} m } {
4716   \begingroup
4717   \stex_reactivate_macro:N \premise
4718   \stex_reactivate_macro:N \conclusion
4719   \__stex_statements_inlineex_args:n{ #1 }
4720   \str_if_empty:NF \sexampleid {
4721     \stex_ref_new_doc_target:n \sexampleid
4722   }
4723   \stex_if_smsmode:TF{
4724     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4725   }{
4726     \seq_clear:N \l_tmpa_seq
4727     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4728       \tl_if_empty:nF{ ##1 }{
4729         \stex_get_symbol:n { ##1 }
4730         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4731           \l_stex_get_symbol_uri_str
4732         }
4733       }
4734     }
4735     \exp_args:Nnx
4736     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{
4737       \str_if_empty:NF \sexamplotype {
4738         \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4739       }
4740       #2
4741       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4742     }
4743   }
4744   \endgroup
4745   \stex_smsmode_do:
4746 }
```

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

33.4 Logical Paragraphs

sparagraph

```

4747 \keys_define:nn { stex / sparagraph } {
4748   id       .str_set_x:N = \sparagraphid ,
4749   title    .tl_set:N    = \l_stex_sparagraph_title_tl ,
4750   type     .str_set_x:N = \sparagraphtype ,
4751   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4752   from     .tl_set:N    = \sparagraphfrom ,
4753   to       .tl_set:N    = \sparagraphto ,
4754   start    .tl_set:N    = \l_stex_sparagraph_start_tl ,
4755   name     .str_set:N   = \sparagraphname
4756 }
4757
4758 \cs_new_protected:Nn \stex_sparagraph_args:n {
4759   \tl_clear:N \l_stex_sparagraph_title_tl
4760   \tl_clear:N \sparagraphfrom
4761   \tl_clear:N \sparagraphto
4762   \tl_clear:N \l_stex_sparagraph_start_tl
4763   \str_clear:N \sparagraphid
4764   \str_clear:N \sparagraphtype
4765   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4766   \str_clear:N \sparagraphname
4767   \keys_set:nn { stex / sparagraph } { #1 }
4768 }
4769 \newif\if@in@omtext\@in@omtextfalse
4770
4771 \NewDocumentEnvironment {sparagraph} { 0{} } {
4772   \stex_sparagraph_args:n { #1 }
4773   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4774     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4775   }{
4776     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4777   }
4778   \@in@omtexttrue
4779   \stex_if_smsmode:F {
4780     \seq_clear:N \l_tmpa_seq
4781     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4782       \tl_if_empty:NF{ ##1 }{
4783         \stex_get_symbol:n { ##1 }
4784         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4785           \l_stex_get_symbol_uri_str
4786         }
4787       }
4788     }
4789     \exp_args:Nnnx
4790     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4791     \str_if_empty:NF \sparagraphtype {
4792       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4793     }
4794     \str_if_empty:NF \sparagraphfrom {
4795       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4796     }
4797     \str_if_empty:NF \sparagraphto {

```

```

4798     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4799 }
4800 \clist_set:No \l_tmpa_clist \sparagraphtype
4801 \tl_clear:N \l_tmpa_tl
4802 \clist_map_inline:Nn \sparagraphtype {
4803     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4804         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4805     }
4806 }
4807 \tl_if_empty:NTF \l_tmpa_tl {
4808     \__stex_statements_sparagraph_start:
4809 }{
4810     \l_tmpa_tl
4811 }
4812 }
4813 \clist_set:No \l_tmpa_clist \sparagraphtype
4814 \str_if_empty:NTF \sparagraphid {
4815     \str_if_empty:NTF \sparagraphname {
4816         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4817             \stex_ref_new_doc_target:n {}
4818         }
4819     } {
4820         \stex_ref_new_doc_target:n {}
4821     }
4822 } {
4823     \stex_ref_new_doc_target:n \sparagraphid
4824 }
4825 \exp_args:NNx
4826 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4827     \clist_map_inline:Nn \__stex_statements_sparagraph_for_clist {
4828         \tl_if_empty:nF{ ##1 }{
4829             \stex_get_symbol:n { ##1 }
4830             \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4831         }
4832     }
4833 }
4834 \stex_smsmode_do:
4835 \ignorespacesandpars
4836 }{
4837     \str_if_empty:NF \sparagraphname {
4838         \stex_symdecl_do:nn{}{\sparagraphname}
4839         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4840     }
4841     \stex_if_smsmode:F {
4842         \clist_set:No \l_tmpa_clist \sparagraphtype
4843         \tl_clear:N \l_tmpa_tl
4844         \clist_map_inline:Nn \l_tmpa_clist {
4845             \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4846                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
4847             }
4848         }
4849         \tl_if_empty:NTF \l_tmpa_tl {
4850             \__stex_statements_sparagraph_end:
4851         }{

```



```

4852     \l_tmpa_tl
4853   }
4854   \end{stex_annotate_env}
4855 }
4856 }

```

\stexpatchparagraph

```

4857
4858 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4859   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4860     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4861       \titleemph{\l_stex_sparagraph_title_tl}:~
4862     }
4863   }{
4864     \titleemph{\l_stex_sparagraph_start_tl}~
4865   }
4866 }
4867 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4868
4869 \newcommand\stexpatchparagraph[3] [] {
4870   \str_set:Nx \l_tmpa_str{ #1 }
4871   \str_if_empty:NTF \l_tmpa_str {
4872     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4873     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4874   }{
4875     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4876     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4877   }
4878 }
4879
4880 \keys_define:nn { stex / inlinepara} {
4881   id      .str_set_x:N = \sparagraphid ,
4882   type    .str_set_x:N = \sparagraphtype ,
4883   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4884   from    .tl_set:N    = \sparagraphfrom ,
4885   to      .tl_set:N    = \sparagraphto ,
4886   name    .str_set:N   = \sparagraphname
4887 }
4888 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4889   \tl_clear:N \sparagraphfrom
4890   \tl_clear:N \sparagraphto
4891   \str_clear:N \sparagraphid
4892   \str_clear:N \sparagraphtype
4893   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4894   \str_clear:N \sparagraphname
4895   \keys_set:nn { stex / inlinepara }{ #1 }
4896 }
4897 \NewDocumentCommand \inlinepara { 0{} m } {
4898   \beginngroup
4899   \__stex_statements_inlinepara_args:n{ #1 }
4900   \clist_set:No \l_tmpa_clist \sparagraphtype
4901   \str_if_empty:NTF \sparagraphid {
4902     \str_if_empty:NTF \sparagraphname {
4903       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{

```

```

4904     \stex_ref_new_doc_target:n {}
4905   }
4906 } {
4907   \stex_ref_new_doc_target:n {}
4908 }
4909 } {
4910   \stex_ref_new_doc_target:n \sparagraphid
4911 }
4912 \stex_if_smsmode:TF{
4913   \str_if_empty:NF \sparagraphname {
4914     \stex_symdecl_do:nn{ }\sparagraphname}
4915   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4916 }
4917 }{
4918   \seq_clear:N \l_tmpa_seq
4919   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4920     \tl_if_empty:nF{ ##1 }{
4921       \stex_get_symbol:n { ##1 }
4922       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4923         \l_stex_get_symbol_uri_str
4924       }
4925     }
4926   }
4927   \exp_args:Nnx
4928   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4929     \str_if_empty:NF \sparagraphtype {
4930       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4931     }
4932     \str_if_empty:NF \sparagraphfrom {
4933       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4934     }
4935     \str_if_empty:NF \sparagraphto {
4936       \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4937     }
4938     \str_if_empty:NF \sparagraphname {
4939       \stex_symdecl_do:nn{ }\sparagraphname}
4940     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4941   }
4942   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4943     \clist_map_inline:Nn \l_tmpa_seq {
4944       \stex_ref_new_sym_target:n {##1}
4945     }
4946   }
4947   #2
4948 }
4949 }
4950 \endgroup
4951 \stex_smsmode_do:
4952 }
4953
4954 (End definition for \stexpatchparagraph. This function is documented on page ??.)
4954 \endpackage

```

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

```
4955 <*package>
4956 <@@=stex_sproof>
4957
4958 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4959
```

34.2 Proofs

We first define some keys for the proof environment.

```
4960 \keys_define:nn { stex / spf } {
4961   id          .str_set_x:N = \spfid,
4962   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
4963   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
4964   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
4965   type        .str_set_x:N = \spftype,
4966   title       .tl_set:N    = \spftitle,
4967   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
4968   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
4969   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
4970 }
4971 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
4972   \str_clear:N \spfid
4973   \tl_clear:N \l__stex_sproof_spf_for_tl
4974   \tl_clear:N \l__stex_sproof_spf_from_tl
4975   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4976   \str_clear:N \spftype
4977   \tl_clear:N \spftitle
4978   \tl_clear:N \l__stex_sproof_spf_continues_tl
4979   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4980 \tl_clear:N \l__stex_sproof_spf_method_tl
4981 \keys_set:nn { stex / spf }{ #1 }
4982 }

```

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

```

4983 \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.

```

4984 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
4985 \cs_new_protected:Npn \sproofnumber {
4986   \int_set:Nn \l_tmpa_int {1}
4987   \bool_while_do:nn {
4988     \int_compare_p:nNn {
4989       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
4990     } > 0
4991   }{
4992     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
4993     \int_incr:N \l_tmpa_int
4994   }
4995 }
4996 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
4997   \int_set:Nn \l_tmpa_int {1}
4998   \bool_while_do:nn {
4999     \int_compare_p:nNn {
5000       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5001     } > 0
5002   }{
5003     \int_incr:N \l_tmpa_int
5004   }
5005   \int_compare:nNnF \l_tmpa_int = 1 {
5006     \int_decr:N \l_tmpa_int
5007   }
5008   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5009     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5010   }

```

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


```

5055 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5056 \input{sproof-finnish.ldf}
5057 }
5058 \clist_if_in:NnT \l_tmpa_clist {french}{
5059 \input{sproof-french.ldf}
5060 }
5061 \clist_if_in:NnT \l_tmpa_clist {russian}{
5062 \input{sproof-russian.ldf}
5063 }
5064 \makeatother
5065 }{}
5066 }

```

spfsketch

```

5067 \newcommand\spfsketch[2] [] {
5068 \beginingroup
5069 \let \premise \stex_proof_premise:
5070 \__stex_sproof_spf_args:n{#1}
5071 \stex_if_smsmode:TF {
5072 \str_if_empty:NF \spfid {
5073 \stex_ref_new_doc_target:n \spfid
5074 }
5075 }{
5076 \seq_clear:N \l_tmpa_seq
5077 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5078 \tl_if_empty:NF{ ##1 }{
5079 \stex_get_symbol:n { ##1 }
5080 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5081 \l_stex_get_symbol_uri_str
5082 }
5083 }
5084 }
5085 \exp_args:Nnx
5086 \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5087 \str_if_empty:NF \spftype {
5088 \stex_annotate_invisible:nnn{type}{\spftype}{}}
5089 }
5090 \clist_set:Nn \l_tmpa_clist \spftype
5091 \tl_set:Nn \l_tmpa_tl {
5092 \titleemph{
5093 \tl_if_empty:NTF \spftitle {
5094 \spf@proofsketch@kw
5095 }{
5096 \spftitle
5097 }
5098 }::~
5099 }
5100 \clist_map_inline:Nn \l_tmpa_clist {
5101 \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5102 \tl_clear:N \l_tmpa_tl
5103 }
5104 }
5105 \str_if_empty:NF \spfid {
5106 \stex_ref_new_doc_target:n \spfid

```

```

5107     }
5108     \l_tmpa_tl #2 \sproofend
5109   }
5110 }
5111 \endgroup
5112 \stex_smsmode_do:
5113 }
5114

```

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

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

```

5115 \newenvironment{spfeq}[2][]{
5116   \__stex_sproof_spf_args:n{#1}
5117   \let \premise \stex_proof_premise:
5118   \stex_if_smsmode:TF {
5119     \str_if_empty:NF \spfid {
5120       \stex_ref_new_doc_target:n \spfid
5121     }
5122   }{
5123     \seq_clear:N \l_tmpa_seq
5124     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5125       \tl_if_empty:nF{ ##1 }{
5126         \stex_get_symbol:n { ##1 }
5127         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5128           \l_stex_get_symbol_uri_str
5129         }
5130       }
5131     }
5132     \exp_args:Nnnx
5133     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5134     \str_if_empty:NF \spftype {
5135       \stex_annotate_invisible:nnn{type}{\spftype}{}
5136     }
5137
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_start:}{
5142         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_spfeq_##1_start:}}
5143       }
5144       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5145         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5146       }
5147     }
5148     \tl_if_empty:NTF \l_tmpa_tl {
5149       \__stex_sproof_spfeq_start:
5150     }{
5151       \l_tmpa_tl
5152     }{-#2}
5153     \str_if_empty:NF \spfid {

```

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

¹⁵EDNOTE: document above

```

5154     \stex_ref_new_doc_target:n \spfid
5155   }
5156   \begin{displaymath}\begin{array}{rc1l}
5157   }
5158   \stex_smsmode_do:
5159 }{
5160   \stex_if_smsmode:F {
5161     \end{array}\end{displaymath}
5162     \clist_set:No \l_tmpa_clist \spftype
5163     \tl_clear:N \l_tmpa_tl
5164     \clist_map_inline:Nn \l_tmpa_clist {
5165       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5166         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5167       }
5168     }
5169     \tl_if_empty:NTF \l_tmpa_tl {
5170       \__stex_sproof_spfeq_end:
5171     }{
5172       \l_tmpa_tl
5173     }
5174     \end{stex_annotate_env}
5175   }
5176 }
5177
5178 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5179   \titleemph{
5180     \tl_if_empty:NTF \spftitle {
5181       \spf@proof@kw
5182     }{
5183       \spftitle
5184     }
5185   }:
5186 }
5187 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5188
5189 \newcommand\stexpatchspfeq[3] [] {
5190   \str_set:Nx \l_tmpa_str{ #1 }
5191   \str_if_empty:NTF \l_tmpa_str {
5192     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5193     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5194   }{
5195     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5196     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5197   }
5198 }
5199

```

(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.

```

5200 \newenvironment{sproof}[2] []{
5201   \let \premise \stex_proof_premise:

```



```

5202 \intarray_gzero:N \l__stex_sproof_counter_intarray
5203 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5204 \__stex_sproof_spf_args:n{#1}
5205 \stex_if_smsmode:TF {
5206   \str_if_empty:NF \spfid {
5207     \stex_ref_new_doc_target:n \spfid
5208   }
5209 }{
5210   \seq_clear:N \l_tmpa_seq
5211   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5212     \tl_if_empty:nF{ ##1 }{
5213       \stex_get_symbol:n { ##1 }
5214       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5215         \l_stex_get_symbol_uri_str
5216       }
5217     }
5218   }
5219   \exp_args:Nnnx
5220   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5221   \str_if_empty:NF \spftype {
5222     \stex_annotate_invisible:nnn{type}{\spftype}{}
5223   }
5224
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_start:}{
5229       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5230     }
5231     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5232       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5233     }
5234   }
5235   \tl_if_empty:NTF \l_tmpa_tl {
5236     \__stex_sproof_sproof_start:
5237   }{
5238     \l_tmpa_tl
5239   }{~#2}
5240   \str_if_empty:NF \spfid {
5241     \stex_ref_new_doc_target:n \spfid
5242   }
5243   \begin{description}
5244 }
5245 \stex_smsmode_do:
5246 }{
5247   \stex_if_smsmode:F{
5248     \end{description}
5249     \clist_set:No \l_tmpa_clist \spftype
5250     \tl_clear:N \l_tmpa_tl
5251     \clist_map_inline:Nn \l_tmpa_clist {
5252       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5253         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5254       }
5255     }

```

```

5256 \tl_if_empty:NTF \l_tmpa_tl {
5257   \__stex_sproof_sproof_end:
5258 }{
5259   \l_tmpa_tl
5260 }
5261 \end{stex_annotate_env}
5262 }
5263 }
5264
5265 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5266   \par\noindent\titleemph{
5267     \tl_if_empty:NTF \spftype {
5268       \spf@proof@kw
5269     }{
5270       \spftype
5271     }
5272   }:
5273 }
5274 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5275
5276 \newcommand\stexpatchsproof[3] [] {
5277   \str_set:Nx \l_tmpa_str{ #1 }
5278   \str_if_empty:NTF \l_tmpa_str {
5279     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5280     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5281   }{
5282     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5283     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5284   }
5285 }

```

\spfidea

```

5286 \newcommand\spfidea[2] []{
5287   \__stex_sproof_spf_args:n{#1}
5288   \titleemph{
5289     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
5290       \l__stex_sproof_spf_type_tl
5291     }:
5292   }~#2
5293   \sproofend
5294 }

```

(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

```

5295 \newenvironment{spfstep}[1] []{
5296   \__stex_sproof_spf_args:n{#1}
5297   \stex_if_smsmode:TF {
5298     \str_if_empty:NF \spfid {

```

```

5299     \stex_ref_new_doc_target:n \spfid
5300   }
5301 }{
5302   \@in@omtexttrue
5303   \seq_clear:N \l_tmpa_seq
5304   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5305     \tl_if_empty:nF{ ##1 }{
5306       \stex_get_symbol:n { ##1 }
5307       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5308         \l_stex_get_symbol_uri_str
5309       }
5310     }
5311   }
5312   \exp_args:Nnnx
5313   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5314   \str_if_empty:NF \spftype {
5315     \stex_annotate_invisible:nnn{type}{\spftype}{ }
5316   }
5317   \clist_set:N \l_tmpa_clist \spftype
5318   \tl_set:Nn \l_tmpa_tl {
5319     \item[\sproofnumber]
5320   }
5321   \clist_map_inline:Nn \l_tmpa_clist {
5322     \exp_args:N \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5323       \tl_clear:N \l_tmpa_tl
5324     }
5325   }
5326   \l_tmpa_tl
5327   \tl_if_empty:NF \spftitle {
5328     {(\titleemph{\spftitle})\enspace}
5329   }
5330   \str_if_empty:NF \spfid {
5331     \stex_ref_new_doc_target:n \spfid
5332   }
5333 }
5334 \__stex_sproof_inc_counter:
5335 \stex_smsmode_do:
5336 \ignorespacesandpars
5337 }{
5338   \stex_if_smsmode:F {
5339     \end{stex_annotate_env}
5340   }
5341 }

```

sproofcomment

```

5342 \newenvironment{sproofcomment}[1][]{
5343   \__stex_sproof_spf_args:n{#1}
5344   \clist_set:N \l_tmpa_clist \spftype
5345   \tl_set:Nn \l_tmpa_tl {
5346     \item[\sproofnumber]
5347   }
5348   \clist_map_inline:Nn \l_tmpa_clist {
5349     \exp_args:N \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5350       \tl_clear:N \l_tmpa_tl

```

```

5351     }
5352   }
5353   \l_tmpa_tl
5354 }{
5355 }

```

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.

```

5356 \newenvironment{subproof}[2][]{
5357   \_stex_sproof_spf_args:n{#1}
5358   \stex_if_smsmode:TF{
5359     \str_if_empty:NF \spfid {
5360       \stex_ref_new_doc_target:n \spfid
5361     }
5362   }{
5363     \seq_clear:N \l_tmpa_seq
5364     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5365       \tl_if_empty:NF{ ##1 }{
5366         \stex_get_symbol:n { ##1 }
5367         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5368           \l_stex_get_symbol_uri_str
5369         }
5370       }
5371     }
5372     \exp_args:Nnnx
5373     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5374     \str_if_empty:NF \spftype {
5375       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5376     }
5377
5378     \clist_set:No \l_tmpa_clist \spftype
5379     \tl_set:Nn \l_tmpa_tl {
5380       \item[\sproofnumber]
5381     }
5382     \clist_map_inline:Nn \l_tmpa_clist {
5383       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5384         \tl_clear:N \l_tmpa_tl
5385       }
5386     }
5387     \l_tmpa_tl
5388     \tl_if_empty:NF \spftitle {
5389       {(\titleemph{\spftitle})\enspace}
5390     }
5391     {~#2}
5392     \str_if_empty:NF \spfid {
5393       \stex_ref_new_doc_target:n \spfid
5394     }
5395   }
5396   \_stex_sproof_add_counter:
5397   \stex_smsmode_do:
5398 }{
5399   \_stex_sproof_remove_counter:

```

```

5400 \stex_sproof_inc_counter:
5401 \stex_if_smsmode:F{
5402   \end{stex_annotate_env}
5403 }
5404 }

```

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

```

5405 \newenvironment{spfcases}[2][]{
5406   \tl_if_empty:nTF{#1}{
5407     \begin{subproof}[method=by-cases]{#2}
5408   }{
5409     \begin{subproof}[#1,method=by-cases]{#2}
5410   }
5411 }{
5412   \end{subproof}
5413 }

```

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

```

5414 \newenvironment{spfcase}[2][]{
5415   \stex_sproof_spf_args:n{#1}
5416   \stex_if_smsmode:TF {
5417     \str_if_empty:NF \spfid {
5418       \stex_ref_new_doc_target:n \spfid
5419     }
5420   }{
5421     \seq_clear:N \l_tmpa_seq
5422     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5423       \tl_if_empty:nF{ ##1 }{
5424         \stex_get_symbol:n { ##1 }
5425         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5426           \l_stex_get_symbol_uri_str
5427         }
5428       }
5429     }
5430     \exp_args:Nnnx
5431     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5432     \str_if_empty:NF \spftype {
5433       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5434     }
5435     \clist_set:Nn \l_tmpa_clist \spftype
5436     \tl_set:Nn \l_tmpa_tl {
5437       \item[\sproofnumber]
5438     }
5439     \clist_map_inline:Nn \l_tmpa_clist {
5440       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5441         \tl_clear:N \l_tmpa_tl
5442       }
5443     }
5444     \l_tmpa_tl
5445     \tl_if_empty:nF{#2}{
5446       \titleemph{#2}:~
5447     }
5448   }

```

```

5449 \stex_sproof_inc_counter:
5450 \stex_smsmode_do:
5451 }{
5452 \stex_if_smsmode:F{
5453 \clist_set:No \l_tmpa_clist \spftype
5454 \tl_set:Nn \l_tmpa_tl{\sproofend}
5455 \clist_map_inline:Nn \l_tmpa_clist {
5456 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5457 \tl_clear:N \l_tmpa_tl
5458 }
5459 }
5460 \l_tmpa_tl
5461 \end{stex_annotate_env}
5462 }
5463 }

```

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

```

5464 \newcommand\spfcasesketch[3][]{
5465 \begin{spfcase}[#1]{#2}#3\end{spfcase}
5466 }

```

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.

```

5467 \keys_define:nn { stex / just }{
5468 id .str_set_x:N = \l__stex_sproof_just_id_str,
5469 method .tl_set:N = \l__stex_sproof_just_method_tl,
5470 premises .tl_set:N = \l__stex_sproof_just_premises_tl,
5471 args .tl_set:N = \l__stex_sproof_just_args_tl
5472 }

```

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

justification

```

5473 \newenvironment{justification}[1]{}{}

```

\premise

```

5474 \newcommand\stex_proof_premise:[2]{}{#2}

```

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

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

```

5475 \newcommand\justarg[2]{}{#2}
5476 \</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.

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

Chapter 35

STEX -Others Implementation

```
5477 <*package>
5478
5479 %%%%%%%%%%% others.dtx %%%%%%%%%%%
5480
5481 <@@=stex_others>
    Warnings and error messages
5482 % None

\MSC Math subject classifier

5483 \NewDocumentCommand \MSC {m} {
5484 % TODO
5485 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5486 \@ifpackageloaded{tikzinput}{
5487 \RequirePackage{stex-tikzinput}
5488 }{}
5489 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
5490 <*package>
5491 <@@=stex_modules>
5492
5493 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5494
5495 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5496 \begingroup
5497 \stex_module_setup:nn{
5498   ns=\c_stex_metatheory_ns_str,
5499   meta=NONE
5500 }{Metatheory}
5501 \stex_reactivate_macro:N \symdecl
5502 \stex_reactivate_macro:N \notation
5503 \stex_reactivate_macro:N \symdef
5504 \ExplSyntaxOff
5505 \csname stex_suppress_html:n\endcsname{
5506   % is-a (a:A, a \in A, a is an A, etc.)
5507   \symdecl{isa}[args=ai]
5508   \notation{isa}[typed]{#1 \comp{:} #2}{##1 \comp, ##2}
5509   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5510   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5511
5512   % bind (\forall, \Pi, \lambda etc.)
5513   \symdecl{bind}[args=Bi]
5514   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
5515   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5516   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
5517
5518   % dummy variable
5519   \symdecl{dummyvar}
5520   \notation{dummyvar}[underscore]{\comp\_}
5521   \notation{dummyvar}[dot]{\comp\cdot}
5522   \notation{dummyvar}[dash]{\comp{\rm --}}
5523
5524   %fromto (function space, Hom-set, implication etc.)
```



```

5525 \symdecl{fromto}[args=ai]
5526 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5527 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5528
5529 % mapto (lambda etc.)
5530 \symdecl{mapto}[args=Bi]
5531 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5532 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5533 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5534
5535 % function/operator application
5536 \symdecl{apply}[args=ia]
5537 \notation{apply}[prec=0;0x\infpref,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5538 \notation{apply}[prec=0;0x\infpref,lambda]{#1 \; #2 }{##1 \; ; ##2}
5539
5540 % ‘type’ of all collections (sets, classes, types, kinds)
5541 \symdecl{collection}
5542 \notation{collection}[U]{\comp{\mathcal{U}}}
5543 \notation{collection}[set]{\comp{\textsf{Set}}}
5544
5545 % collection of propositions/booleans/truth values
5546 \symdecl{prop}[name=proposition]
5547 \notation{prop}[prop]{\comp{\rm prop}}
5548 \notation{prop}[BOOL]{\comp{\rm BOOL}}
5549
5550 % sequences
5551 \symdecl{seqtype}[args=1]
5552 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5553
5554 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5555 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
5556
5557 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,ellipses}}{##1\comp,##2}
5558 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,ellipses,}#2}{##1\comp,##2}
5559 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{\,ellipses,}#2\comp{\,ellipses,}#3}
5560
5561 % letin (‘let’, local definitions, variable substitution)
5562 \symdecl{letin}[args=bii]
5563 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
5564 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5565 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5566
5567 % structures
5568 \symdecl*{module-type}[args=1]
5569 \notation{module-type}{\mathtt{MOD} #1}
5570 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5571 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5572
5573 }
5574 \ExplSyntaxOn
5575 \stex_add_to_current_module:n{
5576   \let\appa\apply
5577   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5578   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5579 \def\livar{\csname sequence-index\endcsname[li]}
5580 \def\uivar{\csname sequence-index\endcsname[ui]}
5581 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5582 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5583 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5584 }
5585 \__stex_modules_end_module:
5586 \endgroup
5587 </package>

```

Chapter 37

Tikzinput Implementation

```
5588 <*package>
5589
5590 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5591
5592 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5593 \RequirePackage{l3keys2e}
5594
5595 \keys_define:nn { tikzinput } {
5596   image .bool_set:N = \c_tikzinput_image_bool,
5597   image .default:n = false ,
5598   unknown .code:n = {}
5599 }
5600
5601 \ProcessKeysOptions { tikzinput }
5602
5603 \bool_if:NTF \c_tikzinput_image_bool {
5604   \RequirePackage{graphicx}
5605
5606   \providecommand\usetikzlibrary[]{}
5607   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5608 }{
5609   \RequirePackage{tikz}
5610   \RequirePackage{standalone}
5611
5612   \newcommand \tikzinput [2] [] {
5613     \setkeys{Gin}{#1}
5614     \ifx \Gin@ewidth \Gin@exclamation
5615       \ifx \Gin@eheight \Gin@exclamation
5616         \input { #2 }
5617       \else
5618         \resizebox{!}{ \Gin@eheight }{
5619           \input { #2 }
5620         }
5621       \fi
5622     \else
5623       \ifx \Gin@eheight \Gin@exclamation
5624         \resizebox{ \Gin@ewidth }{!}{
5625           \input { #2 }
```

```

5626     }
5627     \else
5628         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5629             \input { #2 }
5630         }
5631     \fi
5632 \fi
5633 }
5634 }
5635
5636 \newcommand \ctikzinput [2] [] {
5637     \begin{center}
5638         \tikzinput [ #1 ] { #2 }
5639     \end{center}
5640 }
5641
5642 \@ifpackageloaded{stex}{
5643     \RequirePackage{stex-tikzinput}
5644 }{}
5645
5646 </package>
5647 <*stex>
5648 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5649 \RequirePackage{stex}
5650 \RequirePackage{tikzinput}
5651
5652 \newcommand\mhtikzinput [2] [] {%
5653     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5654     \stex_in_repository:nn\Gin@mhrepos{
5655         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5656     }
5657 }
5658 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5659 </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.

```
5660 \*cls)
5661 \<@@=document_structure>
5662 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5663 \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`

```
5664 \keys_define:nn{ document-structure / pkg }{
5665   class      .str_set_x:N = \c_document_structure_class_str,
5666   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5667   report     .code:n      = {
5668     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5669     \str_set:Nn \c_document_structure_class_str {report}
5670   },
5671   book       .code:n      = {
5672     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5673     \str_set:Nn \c_document_structure_class_str {book}
5674   },
5675   bookpart   .code:n      = {
5676     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5677     \str_set:Nn \c_document_structure_class_str {book}
5678     \str_set:Nn \c_document_structure_topsect_str {chapter}
5679   },
```

```

5680 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5681 unknown     .code:n      = {
5682   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5683 }
5684 }
5685 \ProcessKeysOptions{ document-structure / pkg }
5686 \str_if_empty:NT \c_document_structure_class_str {
5687   \str_set:Nn \c_document_structure_class_str {article}
5688 }
5689 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5690   {\c_document_structure_class_str}
5691

```

38.3 Beefing up the document environment

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

```

5692 \RequirePackage{document-structure}
5693 \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

```

5694 \keys_define:nn { document-structure / document }{
5695   id .str_set_x:N = \c_document_structure_document_id_str
5696 }
5697 \let\__document_structure_orig_document=\document
5698 \renewcommand{\document}[1][]{
5699   \keys_set:nn{ document-structure / document }{ #1 }
5700   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5701   \__document_structure_orig_document
5702 }

```

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

```

5703 }
5704 \</cls>

```

38.4 Implementation: document-structure Package

```

5705 \<*package>
5706 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5707 \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

```

5708
5709 \keys_define:nn{ document-structure / pkg }{
5710   class      .str_set_x:N = \c_document_structure_class_str,
5711   topsect    .str_set_x:N = \c_document_structure_topsect_str,
5712   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5713 }
5714 \ProcessKeysOptions{ document-structure / pkg }
5715 \str_if_empty:NT \c_document_structure_class_str {
5716   \str_set:Nn \c_document_structure_class_str {article}
5717 }
5718 \str_if_empty:NT \c_document_structure_topsect_str {
5719   \str_set:Nn \c_document_structure_topsect_str {section}
5720 }

```

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

```

5721 \RequirePackage{xspace}
5722 \RequirePackage{comment}
5723 \AddToHook{begindocument}{
5724   \ltx@ifpackageloaded{babel}{
5725     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5726     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5727       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5728     }
5729   }{}
5730 }

```

`\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.

```

5731 \int_new:N \l_document_structure_section_level_int
5732 \str_case:VnF \c_document_structure_topsect_str {
5733   {part}}{
5734     \int_set:Nn \l_document_structure_section_level_int {0}
5735   }
5736   {chapter}}{
5737     \int_set:Nn \l_document_structure_section_level_int {1}
5738   }
5739 }{
5740   \str_case:VnF \c_document_structure_class_str {
5741     {book}}{
5742       \int_set:Nn \l_document_structure_section_level_int {0}
5743     }
5744     {report}}{
5745       \int_set:Nn \l_document_structure_section_level_int {0}
5746     }
5747   }{
5748     \int_set:Nn \l_document_structure_section_level_int {2}
5749   }
5750 }

```

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

```
5751 \def\current@section@level{document}%
5752 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5753 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5754 \cs_new_protected:Npn \skipomgroup {
5755   \ifcase\l_document_structure_section_level_int
5756   \or\stepcounter{part}
5757   \or\stepcounter{chapter}
5758   \or\stepcounter{section}
5759   \or\stepcounter{subsection}
5760   \or\stepcounter{subsubsection}
5761   \or\stepcounter{paragraph}
5762   \or\stepcounter{subparagraph}
5763   \fi
5764 }
```

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

`blindomgroup`

```
5765 \newcommand\at@begin@blindomgroup[1]{%
5766 \newenvironment{blindomgroup}
5767 {
5768   \int_incr:N\l_document_structure_section_level_int
5769   \at@begin@blindomgroup\l_document_structure_section_level_int
5770 }{}}
```

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

```
5771 \newcommand\omgroup@nonum[2]{
5772   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5773   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5774 }
```

(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.

```
5775 \newcommand\omgroup@num[2]{
```

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


```

5776 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5777   \@nameuse{#1}{#2}
5778 }{
5779   \cs_if_exist:NTF\rdfmata@sectioning{
5780     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5781   }{
5782     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5783   }
5784 }
5785 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5786 }

```

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

omgroup

```

5787 \keys_define:nn { document-structure / omgroup }{
5788   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5789   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5790   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5791   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5792   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5793   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5794   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5795   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5796   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5797   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5798 }
5799 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5800   \str_clear:N \l__document_structure_omgroup_id_str
5801   \str_clear:N \l__document_structure_omgroup_date_str
5802   \clist_clear:N \l__document_structure_omgroup_creators_clist
5803   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5804   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5805   \tl_clear:N \l__document_structure_omgroup_type_tl
5806   \tl_clear:N \l__document_structure_omgroup_short_tl
5807   \tl_clear:N \l__document_structure_omgroup_display_tl
5808   \tl_clear:N \l__document_structure_omgroup_intro_tl
5809   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5810   \keys_set:nn { document-structure / omgroup } { #1 }
5811 }

```

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.

```

5812 \newif\if@mainmatter\@mainmattertrue
5813 \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.

```

5814 \keys_define:nn { document-structure / sectioning }{
5815   name .str_set_x:N = \l__document_structure_sect_name_str ,
5816   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5817   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5818   clear .default:n = {true} ,
5819   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5820   num      .default:n      = {true}
5821 }
5822 \cs_new_protected:Nn \__document_structure_sect_args:n {
5823   \str_clear:N \l__document_structure_sect_name_str
5824   \str_clear:N \l__document_structure_sect_ref_str
5825   \bool_set_false:N \l__document_structure_sect_clear_bool
5826   \bool_set_false:N \l__document_structure_sect_num_bool
5827   \keys_set:nn { document-structure / sectioning } { #1 }
5828 }
5829 \newcommand\omdoc@sectioning[3][]{
5830   \__document_structure_sect_args:n {#1}
5831   \let\omdoc@sect@name\l__document_structure_sect_name_str
5832   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5833   \if@mainmatter% numbering not overridden by frontmatter, etc.
5834     \bool_if:NTF \l__document_structure_sect_num_bool {
5835       \omgroup@num{#2}{#3}
5836     }{
5837       \omgroup@nonum{#2}{#3}
5838     }
5839     \def\current@section@level{\omdoc@sect@name}
5840   \else
5841     \omgroup@nonum{#2}{#3}
5842   \fi
5843 }% 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.

```

5844 \newcommand\omgroup@redefine@addtocontents[1]{%
5845 %\edef\__document_structureimport{#1}%
5846 %\@for\@I:=\__document_structureimport\do{%
5847 %\edef\@path{\csname module@\@I @path\endcsname}%
5848 %\@ifundefined{tf@toc}\relax%
5849 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5850 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5851 %\def\addcontentsline##1##2##3{%
5852 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5853 %\else% hyperref.sty not loaded
5854 %\def\addcontentsline##1##2##3{%
5855 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5856 %\fi
5857 }% 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.

```

5858 \newenvironment{omgroup}[2][]{% keys, title
5859 {
5860   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

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

```

5861 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5862   \omgroup@redefine@addtocontents{
5863     \@ifundefined{module@id}\used@modules%

```

```

5864      %{\@ifundefined{module@module@id @path}{\used@modules}\module@id}
5865    }
5866  }

```

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

```

5867  \int_incr:N\l_document_structure_section_level_int
5868  \ifcase\l_document_structure_section_level_int
5869    \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5870    \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5871    \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5872    \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5873    \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5874    \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5875    \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5876  \fi
5877  \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5878  \str_if_empty:NF \l__document_structure_omgroup_id_str {
5879    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5880  }
5881  }% for customization
5882  {}

```

and finally, we localize the sections

```

5883  \newcommand\omdoc@part@kw{Part}
5884  \newcommand\omdoc@chapter@kw{Chapter}
5885  \newcommand\omdoc@section@kw{Section}
5886  \newcommand\omdoc@subsection@kw{Subsection}
5887  \newcommand\omdoc@subsubsection@kw{Subsubsection}
5888  \newcommand\omdoc@paragraph@kw{paragraph}
5889  \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`

```

5890  \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).

```

5891  \cs_if_exist:NTF\frontmatter{
5892    \let\__document_structure_orig_frontmatter\frontmatter
5893    \let\frontmatter\relax
5894  }{
5895    \tl_set:Nn\__document_structure_orig_frontmatter{
5896      \clearpage
5897      \@mainmatterfalse
5898      \pagenumbering{roman}

```

```

5899 }
5900 }
5901 \cs_if_exist:NTF\backmatter{
5902   \let\__document_structure_orig_backmatter\backmatter
5903   \let\backmatter\relax
5904 }{
5905   \tl_set:Nn\__document_structure_orig_backmatter{
5906     \clearpage
5907     \@mainmatterfalse
5908     \pagenumbering{roman}
5909   }
5910 }

```

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.

```

5911 \newenvironment{frontmatter}{
5912   \__document_structure_orig_frontmatter
5913 }{
5914   \cs_if_exist:NTF\mainmatter{
5915     \mainmatter
5916   }{
5917     \clearpage
5918     \@mainmattertrue
5919     \pagenumbering{arabic}
5920   }
5921 }

```

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

```

5922 \newenvironment{backmatter}{
5923   \__document_structure_orig_backmatter
5924 }{
5925   \cs_if_exist:NTF\mainmatter{
5926     \mainmatter
5927   }{
5928     \clearpage
5929     \@mainmattertrue
5930     \pagenumbering{arabic}
5931   }
5932 }

```

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

```

5933 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5934 \def \c__document_structure_document_str{document}
5935 \newcommand\afterprematurestop{}
5936 \def\prematurestop@endomgroup{
5937   \unless\ifx\@currenvir\c__document_structure_document_str
5938     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
5939     \expandafter\prematurestop@endomgroup

```

```

5940 \fi
5941 }
5942 \providecommand\prematurestop{
5943 \message{Stopping~sTeX~processing~prematurely}
5944 \prematurestop@endomgroup
5945 \afterprematurestop
5946 \end{document}
5947 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5948 \RequirePackage{etoolbox}
5949 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5950 \newrobustcmd\useSGvar[1]{%
5951 \@ifundefined{sTeX@Gvar@#1}
5952 {\PackageError{document-structure}
5953 {The sTeX Global variable #1 is undefined}
5954 {set it with \protect\setSGvar}}
5955 \@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.

```

5956 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5957 \@ifundefined{sTeX@Gvar@#1}
5958 {\PackageError{document-structure}
5959 {The sTeX Global variable #1 is undefined}
5960 {set it with \protect\setSGvar}}
5961 {\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.

```
5962 \*cls)
5963 \@@=notesslides)
5964 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5965 \RequirePackage{13keys2e}
5966
5967 \keys_define:nn{notesslides / cls}{
5968   class .code:n = {
5969     \PassOptionsToClass{\CurrentOption}{document-structure}
5970     \str_if_eq:nnT{#1}{book}{
5971       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5972     }
5973     \str_if_eq:nnT{#1}{report}{
5974       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5975     }
5976   },
5977   notes .bool_set:N = \c__notesslides_notes_bool ,
5978   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5979   unknown .code:n = {
5980     \PassOptionsToClass{\CurrentOption}{document-structure}
5981     \PassOptionsToClass{\CurrentOption}{beamer}
5982     \PassOptionsToPackage{\CurrentOption}{notesslides}
5983   }
5984 }
5985 \ProcessKeysOptions{ notesslides / cls }
5986 \bool_if:NTF \c__notesslides_notes_bool {
5987   \PassOptionsToPackage{notes=true}{notesslides}
5988 }{
5989   \PassOptionsToPackage{notes=false}{notesslides}
5990 }
5991 \</cls)
```

now we do the same for the notesslides package.

```

5992 <*package>
5993 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5994 \RequirePackage{13keys2e}
5995
5996 \keys_define:nn{notesslides / pkg}{
5997   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5998   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5999   notes        .bool_set:N = \c__notesslides_notes_bool ,
6000   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6001   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6002   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
6003   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6004   noproblems   .bool_set:N = \c__notesslides_noproblems_bool,
6005   unknown      .code:n      = {
6006     \PassOptionsToClass{\CurrentOption}{stex}
6007     \PassOptionsToClass{\CurrentOption}{tikzinput}
6008   }
6009 }
6010 \ProcessKeysOptions{ notesslides / pkg }
6011 \newif\ifnotes
6012 \bool_if:NTF \c__notesslides_notes_bool {
6013   \notesttrue
6014 }{
6015   \notesfalse
6016 }
6017

```

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

```

6018 \str_if_empty:NTF \c__notesslides_topsect_str {
6019   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6020 }{
6021   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6022 }
6023 </package>

```

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

```

6024 <*cls>
6025 \bool_if:NTF \c__notesslides_notes_bool {
6026   \LoadClass{document-structure}
6027 }{
6028   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6029   \newcounter{Item}
6030   \newcounter{paragraph}
6031   \newcounter{subparagraph}
6032   \newcounter{Hfootnote}
6033   \RequirePackage{document-structure}
6034 }

```

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

```

6035 \RequirePackage{notesslides}
6036 </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.

```

6037 \*package>
6038 \bool_if:NT \c__notesslides_notes_bool {
6039   \RequirePackage{a4wide}
6040   \RequirePackage{marginnote}
6041   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6042   \RequirePackage{mdframed}
6043   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6044   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6045 }
6046 \RequirePackage{stex-tikzinput}
6047 \RequirePackage{etoolbox}
6048 \RequirePackage{amssymb}
6049 \RequirePackage{amsmath}
6050 \RequirePackage{comment}
6051 \RequirePackage{textcomp}
6052 \RequirePackage{url}
6053 \RequirePackage{graphicx}
6054 \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

```

6055 \bool_if:NT \c__notesslides_notes_bool {
6056   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
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}

```

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}{\ignorespaces}
6063 }{
6064   \excludecomment{note}
6065 }

```

¹⁹EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
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{framenum}{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 }

```

(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 }

```

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

nomgroup

```

6151 \bool_if:NTF \c__notesslides_notes_bool {
6152   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
6153 }{
6154   \excludecomment{nomgroup}
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}{\TeX}
6194 \newrobustcmd{\setslidelogo}[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]{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
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

```

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

```

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}{dangerous-bend}
6290 \def\smalltextwarning{
6291     \pgfuseimage{miko@small@dbend}
6292     \xspace
6293 }
6294 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

6295 \newrobustcmd\textwarning{
6296   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6297   \xspace
6298 }
6299 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{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 choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

6339 \renewenvironment{omgroup}[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{subsubsection}}
6366           \def\currentsectionlevel{\omdoc@subsubsection@kw}
6367         \or
6368           \stepcounter{paragraph}
6369           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
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\hspace}
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       \linewidth\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   \excludacomment{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{omgroup}[#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{omgroup}
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}{2019/03/20}{1.3}{Semantic Markup for Problems}
6466 \RequirePackage{l3keys2e}
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}{2019/03/20}{1.1}{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}{2019/03/20}{1.1}{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_inclasssign_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_inclasssign_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_inclasssign_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_inclasssign_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{omgroup}[loadmodules]{\ass@title}
7092 }{
7093 \begin{omgroup}{\ass@title}
7094 }
7095 }{
7096 \end{omgroup}
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}

```