

The sTeX3 Package *

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

2022-02-14

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 2022-02-14)

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 Semantic Macros	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
10	sTeX-MathHub	22
10.1	Macros and Environments	22
10.1.1	Files, Paths, URIs	22
10.1.2	MathHub Archives	23

11	sTeX-References	25
11.1	Macros and Environments	25
12	sTeX-Modules	26
12.1	Macros and Environments	26
12.1.1	The <code>module</code> -environment	28
13	sTeX-Module Inheritance	31
13.1	Macros and Environments	31
13.1.1	SMS Mode	31
13.1.2	Imports and Inheritance	32
14	sTeX-Symbols	35
14.1	Macros and Environments	35
15	sTeX-Terms	38
15.1	Macros and Environments	38
16	sTeX-Structural Features	41
16.1	Macros and Environments	41
16.1.1	Structures	41
17	sTeX-Statements	42
17.1	Macros and Environments	42
18	sTeX-Proofs: Structural Markup for Proofs	43
18.1	Introduction	45
18.2	The User Interface	46
18.2.1	Package Options	46
18.2.2	Proofs and Proof steps	46
18.2.3	Justifications	46
18.2.4	Proof Structure	47
18.2.5	Proof End Markers	48
18.2.6	Configuration of the Presentation	48
18.3	Limitations	48
19	sTeX-Metatheory	50
19.1	Symbols	50
III	Extensions	51
20	Tikzinput	52
20.1	Macros and Environments	52

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

25	STeX-Basics Implementation	72
25.1	The STeXDocument Class	72
25.2	Preliminaries	72
25.3	Messages and logging	73
25.4	Persistence	74
25.5	HTML Annotations	74
25.6	Languages	77
25.7	Activating/Deactivating Macros	78
26	STeX-MathHub Implementation	80
26.1	Generic Path Handling	80
26.2	PWD and kpsewhich	82
26.3	File Hooks and Tracking	83
26.4	MathHub Repositories	84
27	STeX-References Implementation	92
27.1	Document URIs and URLs	92
27.2	Setting Reference Targets	94
27.3	Using References	95
28	STeX-Modules Implementation	98
28.1	The module environment	101
28.2	Invoking modules	107
29	STeX-Module Inheritance Implementation	109
29.1	SMS Mode	109
29.2	Inheritance	112
30	STeX-Symbols Implementation	117
30.1	Symbol Declarations	117
30.2	Notations	124
31	STeX-Terms Implementation	133
31.1	Symbol Invocations	133
31.2	Terms	136
31.3	Notation Components	142
32	STeX-Structural Features Implementation	145
32.1	Imports with modification	145
32.2	The feature environment	152
32.3	Features	153
33	STeX-Statements Implementation	159
33.1	Definitions	159
33.2	Assertions	164
33.3	Examples	167
33.4	Logical Paragraphs	169

34 The Implementation	174
34.1 Package Options	174
34.2 Proofs	174
34.3 Justifications	180
35 \TeX-Others Implementation	182
36 \TeX-Metatheory Implementation	183
37 Tikzinput Implementation	186
38 document-structure.sty Implementation	188
38.1 The document-structure Class	188
38.2 Class Options	188
38.3 Beefing up the <code>document</code> environment	189
38.4 Implementation: document-structure Package	189
38.5 Package Options	189
38.6 Document Structure	191
38.7 Front and Backmatter	194
38.8 Global Variables	196
39 NotesSlides – Implementation	197
39.1 Class and Package Options	197
39.2 Notes and Slides	199
39.3 Header and Footer Lines	203
39.4 Frame Images	204
39.5 Colors and Highlighting	205
39.6 Sectioning	206
39.7 Excursions	208
40 The Implementation	210
40.1 Package Options	210
40.2 Problems and Solutions	211
40.3 Multiple Choice Blocks	217
40.4 Including Problems	218
40.5 Reporting Metadata	219
41 Implementation: The hwexam Class	221
41.1 Class Options	221
42 Implementation: The hwexam Package	223
42.1 Package Options	223
42.2 Assignments	224
42.3 Including Assignments	227
42.4 Typesetting Exams	228
42.5 Leftovers	230

Part I
Manual

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

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

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

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

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

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

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

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

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

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

2.2 A First s_TE_X Document

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

The document we will consider is the following:

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

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

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

Compiling this document with **pdflatex** should yield the output

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

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

\usemodule

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

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

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

³EdNOTE: somewhere later

`\symref`
`\symname`

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

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

`\importmodule`

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

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

Chapter 3

Using Semantic Macros

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`: 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

5.1 Advanced Structuring Mechanisms

Given modules:

Example 1

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

Module 1:
Module 2:
Module 3:

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 2

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

Test: $\rtimes a{\rplus c}{\rtimes de}$
\end{smodule}
```

Module 4:
Test: $a \cdot (c + d \cdot e)$

TODO: explain donotclone

Example 3

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

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

Module 5:

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`
`\stex`

Both print this \TeX logo.

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

Example 4

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

ab

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

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

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

Example 5

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

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

.

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

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

Example 6

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

$a * b$ 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 7

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

Multiplying again by b yields...

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

Example 8

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

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

⁴EdNOTE: TODO

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

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

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

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

Module 7: 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 11

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

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

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

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

Example 12

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

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

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

5 6

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

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

Precedences

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

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

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

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

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

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

For example:

Module 8:

Example 13

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

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

8.1.2 Archives and Imports

Namespaces

Ideally, \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

Both the sTeX package and class offer the following package options:

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

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 (see ???).

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

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 {$\langle log-prefix \rangle$} {$\langle message \rangle$}</code>
	Logs $\langle message \rangle$, if the package option debug contains $\langle log-prefix \rangle$.

<code>\stex_add_to_sms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
---------------------------------	--

<code>\if@latexml</code>	L ^A T _E X2e and L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</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 $\langle content \rangle$ 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> $\langle content \rangle$ <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
--------------------------------	---

<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`.

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

Makes the macro $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the $\langle begin \rangle$ -code of the associated environments.

<code>\MSC</code>	<code>\MSC{⟨msc⟩}</code>
-------------------	--------------------------

Designates the *math subject classifier* of the current module / file.

Chapter 10

sTEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and 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> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$
<code>\stex_path_from_string:(NV cn cV)</code>	

turns the $\langle string \rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle path-variable \rangle$. Also applies `\stex_path_canonicalize:N`.

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	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</code>	\underline{NTF} \star

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

<code>\c_stex_pwd_seq</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>\c_stex_pwd_str</code>	
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

`\g_stex_currentfile_seq`

The file being currently processed (respecting `\input` etc.)

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
.././aaa & \cpath@print{.././aaa} & & .././aaa \\
aaa/bbb & \cpath@print{aaa/bbb} & & aaa/bbb \\
aaa/. & \cpath@print{aaa/.} & & \\
.././aaa/bbb & \cpath@print{.././aaa/bbb} & & .././aaa/bbb \\
../aaa/./bbb & \cpath@print{../aaa/./bbb} & & ../bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & & ../aaa/bbb \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/ddd \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/bbb/ddd \\
./ & \cpath@print{./} & & \\
aaa/bbb/./.. & \cpath@print{aaa/bbb/./..} & & \\
\end{tabular}
\end{center}
```

path	canonicalized path	expected
aaa	aaa	aaa
.././aaa	.././aaa	.././aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/.		
.././aaa/bbb	.././aaa/bbb	.././aaa/bbb
../aaa/./bbb	../bbb	../bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb/./ddd	aaa/ddd	aaa/ddd
aaa/bbb/./ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb/./..		

10.1.2 MathHub Archives

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

We determine the path to the local MathHub folder via one of three means, in order of precedence:

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.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; currently not in use).

<hr/> <hr/> <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 MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <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.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<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>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<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 (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \
narr:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \
ns:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \
deps:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff

```

```

id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

All information of a module is stored as a property list. `\l_stex_current_module_str` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

```
\stex_if_in_module_p: * Conditional for whether we are currently in a module
\stex_if_in_module:TF *
```

```
\stex_if_module_exists_p:n *
\stex_if_module_exists:nTF *
```

Conditional for whether a module with the provided URI is already known.

```
\stex_add_to_current_module:n
\STEXexport
```

Adds the provided tokens to the `content` field of the current module.

```
\stex_add_constant_to_current_module:n
```

Adds the declaration with the provided name to the `constants` field of the current module.

```
\stex_add_import_to_current_module:n
```

Adds the module with the provided full URI to the `imports` field of the current module.

```
\stex_modules_compute_namespace:nN \stex_modules_compute_namespace:nN
{\<namespace>}{\<path>}
```

Computes the namespace for file `<path>` in repository with namespace `<namespace>` as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

```
\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff
```

```

Namespace 1:
file://stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

12.1.1 The module-environment

module `\begin{module}[\langle options \rangle]{\langle name \rangle}`
 Opens a new module with name $\langle name \rangle$.
 TODO document options.

`\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.

`\stex_modules_heading:` Takes care of the module header, if the `showmods` package option is true. This macro can be overridden for customization.

@module `\begin{@module}[\langle options \rangle]{\langle name \rangle}`
 Core functionality of the `module-environment` without a header.

Test 4

```

\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{smodule}{Foo}
Module-path:-
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { meta }\\
\end{smodule}
\ExplSyntaxOff

```

```

Module 9:  Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

Test 5

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{smodule}[title=Foo Bar]{Bar}
Module-path:-
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { meta }\\
\end{smodule}
\ExplSyntaxOff
```

Module 10: FooBar Module path: <http://mathhub.info/tests/Foo/Bar/Foo?Bar>
Language:
Signature:
Metatheory:

\STEXModule \STEXModule {<fragment>}

Attempts to find a module whose URI ends with <fragment> 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.

Test 6

```
\begin{smodule}{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\\
\STEXModule{STEXModuleTest1}?{foo}[\comp{foo1}]\\
\STEXModule{STEXModuleTest2}?{foo}[\comp{foo2}]\\
\STEXModule{STEXModuleTest3}?{foo}[\comp{foo3}]\\
\end{smodule}
```

Module 11:
Module 12:
Module 13: file://stextest?STEXModuleTest1
file://stextest?STEXModuleTest2
file://stextest?STEXModuleTest3
foo1
foo2
foo3

`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists:

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

```
Module 14:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
               Meaning: >macro:->\protect \bar <
Module 15:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 16:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

`\usemodule`

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

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

Test 9

```

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

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

```

Module 17:
Module 18: Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}<
Module 19: Meaning: >undefined<
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory,file://stextest?UseTest3,file://stextest?UseTest2>
All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dummyvar>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <file://stextest?UseTest2?bar>

Test 10

```

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

```

Circular dependencies:
Module 20: >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

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

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

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

Chapter 14

TeX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

<code>\symdecl</code>	<code>\symdecl[$\langle args \rangle$]{$\langle macroname \rangle$}</code>
-----------------------	--

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

```
Module 21:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

Module 22:

`\symdef`

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

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

Test 13

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

Module 23: $a + b + c$

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

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

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

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

`\withbrackets`

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

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

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

Test 14

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

Module 24: $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 15

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

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

Module 25: $\langle a \mid [b;c;d;e;f]^g \rangle$ and $\langle a \mid [b;c]^g \rangle$ and $\langle a \mid [b]^c \rangle$

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

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

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

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{smodule}
```

Module 26:

```
some a and some b and also some c here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a
```

`\stex_highlight_term:nn`

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

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

`\comp`
`\compemph`
`\compemph@uri`
`\defemph`
`\defemph@uri`
`\symrefemph`
`\symrefemph@uri`

`\comp{<args>}`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 16

ST_EX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

TeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

18.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

18.2 The User Interface

18.2.1 Package Options

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

18.2.2 Proofs and Proof steps

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

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

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

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

18.2.3 Justifications

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

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

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

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

P.1 For the induction we have to consider the following cases:

P.1.1 $n = 1$: then we compute $1 = 1^2$ □

P.1.1 $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

P.1.1 $n > 1$:

P.1.1.1 Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.

P.1.1.1 We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

P.1.1.1 We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum

P.1.1.1 Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

P.1.1.1 We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □

P.1.1 We have considered all the cases, so we have proven the assertion. □

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

18.2.4 Proof Structure

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

18.2.5 Proof End Markers

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

$$\backslash\text{proofend}$$

$$\backslash\text{sProofEndSymbol}$$

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

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

18.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

$$\backslash\text{pstlabelstyle}$$

`\pstlabelstyle{<style>}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@<style>` that takes 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 2 for examples.

style	example	configuration macro
<code>long</code>	<code>0.8.1.5</code>	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}\#2}</code>
<code>angles</code>	<code>>>>5</code>	<code>\def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}\#2}</code>
<code>short</code>	<code>5</code>	<code>\def\pst@make@label@short#1#2{\#2}</code>
<code>empty</code>		<code>\def\pst@make@label@empty#1#2{}</code>

Figure 2: Configuration Proof Step Label Styles

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 \LaTeX issue tracker at [\[sTeX\]](#).

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

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

This package supplies an infrastructure for writing OMDOC documents in \LaTeX . 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 \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX 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
	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>id</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>creators</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>contributors</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>short</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>
<code>loadmodules</code>	key it needs no value. For instance we would have

```

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

```

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

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

`blindomgroup`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

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

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

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

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

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

21.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets \LaTeX ML generate the correct reference.
`\STRcopy`
`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.¹⁰

21.2.5 Global Variables

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

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

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

21.2.6 Colors

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

21.3 Limitations

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

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

Chapter 22

NotesSlides – Slides and Course Notes

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

22.1 Introduction

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

22.2 The User Interface

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the \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=elefants,pts=10,min=2,title=Fitting Elefants]
    How many Elefants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elefants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

23.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

23.2.4 Including Problems

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

23.2.5 Reporting Metadata

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

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

23.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 24

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

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

Contents

24.1 Introduction

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

24.2 The User Interface

24.2.1 Package and Class Options

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

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

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

24.2.2 Assignments

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

24.2.3 Typesetting Exams

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

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

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

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

24.2.4 Including Assignments

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

24.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 25

ST_EX -Basics Implementation

25.1 The ST_EXDocument Class

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

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

25.2 Preliminaries

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

Package options:

```

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

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

```

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

25.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

25.4 Persistence

77 `<@@=stex_persist>`

`\c__stex_persist_sms_iow` File variable used for the sms-File

```

78 \iow_new:N \c__stex_persist_sms_iow
79 \AddToHook{begindocument}{
80   \bool_if:NTF \c_stex_persist_mode_bool {
81     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
82   } {
83     % \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
84   }
85 }
86 \AddToHook{enddocument}{
87   \bool_if:NF \c_stex_persist_mode_bool {
88     % \iow_close:N \c__stex_persist_sms_iow
89   }
90 }

```

(End definition for `\c__stex_persist_sms_iow`.)

`\stex_add_to_sms:n` Adds the provided code to the .sms-file of the document.

```

91 \cs_new_protected:Nn \stex_add_to_sms:n {
92   \bool_if:NF \c_stex_persist_mode_bool {
93     % \iow_now:Nn \c__stex_persist_sms_iow { #1 }
94   }
95 }

```

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

25.5 HTML Annotations

96 `<@@=stex_annotate>`
97 `\RequirePackage{rustex}`

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

```

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

```

`\if@latexml` Conditionals for L^AT_EX_ML:

```

\latexml_if_p:
\latexml_if:TF
99 \ifcsname if@latexml\endcsname\else

```



```

100 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
101 \fi
102
103 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
104   \if@latexml
105     \prg_return_true:
106   \else:
107     \prg_return_false:
108   \fi:
109 }

```

(End definition for `\if@latexml` and `\latexml_if:TF`. These functions are documented on page 20.)

`\l__stex_annotate_arg_tl` Used by annotation macros to ensure that the HTML output to annotate is not empty.
`\c__stex_annotate_emptyarg_tl`

```

110 \tl_new:N \l__stex_annotate_arg_tl
111 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
112   \rustex_if:TF {
113     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
114   }{-}
115 }

```

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

`_stex_annotate_checkempty:n`

```

116 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
117   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
118   \tl_if_empty:NT \l__stex_annotate_arg_tl {
119     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
120   }
121 }

```

(End definition for `_stex_annotate_checkempty:n`.)

`\l_stex_html_do_output_bool` Whether to (locally) produce HTML output

```

\stex_if_do_html:
122 \bool_new:N \l_stex_html_do_output_bool
123 \bool_set_true:N \l_stex_html_do_output_bool
124 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
125   \bool_if:nTF \l_stex_html_do_output_bool
126     \prg_return_true: \prg_return_false:
127 }

```

(End definition for `\l_stex_html_do_output_bool` and `\stex_if_do_html:`. These functions are documented on page ??.)

`\stex_suppress_html:n` Whether to (locally) produce HTML output

```

128 \cs_new_protected:Nn \stex_suppress_html:n {
129   \exp_args:Nne \use:nn {
130     \bool_set_false:N \l_stex_html_do_output_bool
131     #1
132   }{
133     \stex_if_do_html:T {
134       \bool_set_true:N \l_stex_html_do_output_bool
135     }
136   }
137 }

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EXML, R_US_TE_X, p_DF_LA_TE_X).

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_EXML-implementations resort to perl bindings.

```

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

```

```

186   }{
187     \par\rustex_annotate_HTML_end:
188   }
189 }{
190   \latexml_if:TF {
191     \cs_new_protected:Nn \stex_annotate:nnn {
192       \__stex_annotate_checkempty:n { #3 }
193       \mode_if_math:TF {
194         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
195           \tl_use:N \l__stex_annotate_arg_tl
196         }
197       }{
198         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
199           \tl_use:N \l__stex_annotate_arg_tl
200         }
201       }
202     }
203     \cs_new_protected:Nn \stex_annotate_invisible:n {
204       \__stex_annotate_checkempty:n { #1 }
205       \mode_if_math:TF {
206         \cs:w latexml@invisible@math\cs_end:{
207           \tl_use:N \l__stex_annotate_arg_tl
208         }
209       } {
210         \cs:w latexml@invisible@text\cs_end:{
211           \tl_use:N \l__stex_annotate_arg_tl
212         }
213       }
214     }
215     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
216       \__stex_annotate_checkempty:n { #3 }
217       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
218         \tl_use:N \l__stex_annotate_arg_tl
219       }
220     }
221     \NewDocumentEnvironment{stex_annotate_env} { m m } {
222       \par\begin{latexml@annotateenv}{#1}{#2}
223     }{
224       \par\end{latexml@annotateenv}
225     }
226   }{
227     \cs_new_protected:Nn \stex_annotate:nnn {#3}
228     \cs_new_protected:Nn \stex_annotate_invisible:n {}
229     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
230     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
231   }
232 }

```

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

25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

We store language abbreviations in two (mutually inverse) property lists:

```

234 \prop_const_from_keyval:Nn \c_stex_languages_prop {
235   en = english ,
236   de = ngerman ,
237   ar = arabic ,
238   bg = bulgarian ,
239   ru = russian ,
240   fi = finnish ,
241   ro = romanian ,
242   tr = turkish ,
243   fr = french
244 }
245
246 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
247   english   = en ,
248   ngerman   = de ,
249   arabic    = ar ,
250   bulgarian = bg ,
251   russian   = ru ,
252   finnish   = fi ,
253   romanian  = ro ,
254   turkish   = tr ,
255   french    = fr
256 }
257 % todo: chinese simplified (zhs)
258 %       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:

```

259 \clist_if_empty:NF \c_stex_languages_clist {
260   \clist_clear:N \l_tmpa_clist
261   \clist_map_inline:Nn \c_stex_languages_clist {
262     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
263       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
264     } {
265       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
266     }
267   }
268   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
269   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
270 }

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

271 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
272   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
273   \def#1{
274     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
275   }
276 }

```

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

`\stex_reactivate_macro:N`

```

277 \cs_new_protected:Nn \stex_reactivate_macro:N {
278   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
279 }

```

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

`\stex_do_aftergroup:nn`

```

280 <@@=stex_aftergroup>
281 \tl_new:N \l__stex_aftergroup_tl
282 \cs_new_protected:Nn \stex_do_aftergroup:n {
283   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
284     #1
285   }{
286     #1
287     \expandafter \tl_gset:Nn \expandafter \l__stex_aftergroup_tl \expandafter { \l__stex_aft
288     \aftergroup\__stex_aftergroup_do:
289   }
290 }
291 \cs_new_protected:Nn \__stex_aftergroup_do: {
292   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
293     \l__stex_aftergroup_tl
294     \tl_clear:N \l__stex_aftergroup_tl
295   }{
296     \l__stex_aftergroup_tl
297     \aftergroup\__stex_aftergroup_do:
298   }
299 }

```

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

```

300 </package>

```

Chapter 26

STEX -MathHub Implementation

```
301 <*package>
302
303 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
304
305 <@@=stex_path>
306
307 Warnings and error messages
308 \msg_new:nnn{stex}{error/norepository}{
309   No~archive~#1~found~in~#2
310 }
311 \msg_new:nnn{stex}{error/notinarchive}{
312   Not~currently~in~an~archive,~but~\detokenize{#1}~
313   needs~one!
314 }
315 \msg_new:nnn{stex}{error/nofile}{
316   \detokenize{#1}~could~not~find~file~#2
317 }
318 \msg_new:nnn{stex}{error/twofiles}{
319   \detokenize{#1}~found~two~candidates~for~#2
320 }
```

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
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
319 \cs_new_protected:Nn \stex_path_from_string:Nn {
320   \str_set:Nx \l_tmpa_str { #2 }
321   \str_if_empty:NTF \l_tmpa_str {
322     \seq_clear:N #1
323   }{
324     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
325     \sys_if_platform_windows:T{
326       \seq_clear:N \l_tmpa_tl
```

```

327     \seq_map_inline:Nn #1 {
328       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
329       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
330     }
331     \seq_set_eq:NN #1 \l_tmpa_tl
332   }
333   \stex_path_canonicalize:N #1
334 }
335 }
336 \cs_generate_variant:Nn \stex_path_from_string:Nn
337 { NV, cn, cV }

```

(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`

```

338 \cs_new_protected:Nn \stex_path_to_string:NN {
339   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
340 }
341
342 \cs_new:Nn \stex_path_to_string:N {
343   \seq_use:Nn #1 /
344 }

```

(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`

```

345 \str_const:Nn \c__stex_path_dot_str {.}
346 \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.

```

347 \cs_new_protected:Nn \stex_path_canonicalize:N {
348   \seq_if_empty:NF #1 {
349     \seq_clear:N \l_tmpa_seq
350     \seq_get_left:NN #1 \l_tmpa_tl
351     \str_if_empty:NT \l_tmpa_tl {
352       \seq_put_right:Nn \l_tmpa_seq {}
353     }
354     \seq_map_inline:Nn #1 {
355       \str_set:Nn \l_tmpa_tl { ##1 }
356       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
357         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
358           \seq_if_empty:NNTF \l_tmpa_seq {
359             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
360               \c__stex_path_up_str
361             }
362           }{
363             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
364             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
365               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
366                 \c__stex_path_up_str
367               }

```

```

368         }{
369         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
370         }
371     }
372     }{
373     \str_if_empty:NF \l_tmpa_tl {
374     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
375     }
376     }
377 }
378 }
379 \seq_gset_eq:NN #1 \l_tmpa_seq
380 }
381 }

```

(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}`

```

382 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
383   \seq_if_empty:NTF #1 {
384     \prg_return_false:
385   }{
386     \seq_get_left:NN #1 \l_tmpa_tl
387     \str_if_empty:NTF \l_tmpa_tl {
388       \prg_return_true:
389     }{
390       \prg_return_false:
391     }
392   }
393 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

394 \str_new:N\l_stex_kpsewhich_return_str
395 \cs_new_protected:Nn \stex_kpsewhich:n {
396   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
397   \exp_args:NNo \str_set:Nn \l_stex_kpsewhich_return_str{\l_tmpa_tl}
398   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
399 }

```

(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`

```

400 \sys_if_platform_windows:TF{
401   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
402 }{
403   \stex_kpsewhich:n{-var-value~PWD}
404 }
405

```



```

406 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
407 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
408 \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

```

409 <@@=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

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

411 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
412 \stex_path_from_string:Nn \c_stex_mainfile_seq
413 \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` Hooks for file inputs that push/pop `\g_stex_files_stack` to update `\c_stex_mainfile_seq`.

```

414 \seq_gclear_new:N\g_stex_currentfile_seq
415 \cs_new_protected:Nn \stex_filestack_push:n {
416   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
417   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
418     \stex_path_from_string:Nn\g_stex_currentfile_seq{
419       \c_stex_pwd_str/#1
420     }
421   }
422   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
423   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
424 }
425 \cs_new_protected:Nn \stex_filestack_pop: {
426   \seq_if_empty:NF\g_stex_files_stack{
427     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
428   }
429   \seq_if_empty:NTF\g_stex_files_stack{
430     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
431   }{
432     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
433     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
434   }
435 }
436

```

```

437 \AddToHook{file/before}{
438   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
439 }
440 \AddToHook{file/after}{
441   \stex_filestack_pop:
442 }

```

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

26.4 MathHub Repositories

```

443 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
444 \str_if_empty:NTF\mathhub{
445   \stex_kpsewhich:n{-var-value~MATHHUB}
446   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
447
448   \str_if_empty:NTF\c_stex_mathhub_str{
449     \msg_warning:nn{stex}{warning/nomathhub}
450   }{
451     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
452     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
453   }
454 }{
455   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
456   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
457     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
458       \c_stex_pwd_str/\mathhub
459     }
460   }
461   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
462   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463 }

```

(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
464 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
465   \str_set:Nx \l_tmpa_str { #1 }
466   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
467     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
468     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
469     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
470     \__stex_mathhub_find_manifest:N \l_tmpa_seq
471     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
472       \msg_error:nnxx{stex}{error/norepository}{#1}{
473         \stex_path_to_string:N \c_stex_mathhub_str
474       }
475     } {
476       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
477     }
478   }
479 }

```

(End definition for _stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

480 \str_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:

```

481 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
482   \seq_set_eq:NN\l_tmpa_seq #1
483   \bool_set_true:N\l_tmpa_bool
484   \bool_while_do:Nn \l_tmpa_bool {
485     \seq_if_empty:NTF \l_tmpa_seq {
486       \bool_set_false:N\l_tmpa_bool
487     }{
488       \file_if_exist:nTF{
489         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
490       }{
491         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
492         \bool_set_false:N\l_tmpa_bool
493       }{
494         \file_if_exist:nTF{
495           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
496         }{
497           \seq_put_right:Nn\l_tmpa_seq{META-INF}
498           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
499           \bool_set_false:N\l_tmpa_bool
500         }{
501           \file_if_exist:nTF{
502             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
503           }{
504             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
505             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
506             \bool_set_false:N\l_tmpa_bool
507           }{
508             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
509           }
510         }
511       }
512     }
513   }
514   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
515 }

```

(End definition for _stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior File variable used for MANIFEST-files

516 \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:

```

517 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
518   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
519   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
520   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
521     \str_set:Nn \l_tmpa_str {##1}
522     \exp_args:NNoo \seq_set_split:Nnn
523       \l_tmpb_seq \c_colon_str \l_tmpa_str
524     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
525       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
526         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
527       }
528       \exp_args:No \str_case:nnTF \l_tmpa_tl {
529         {id} {
530           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531             { id } \l_tmpb_tl
532         }
533         {narration-base} {
534           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535             { narr } \l_tmpb_tl
536         }
537         {url-base} {
538           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539             { docurl } \l_tmpb_tl
540         }
541         {source-base} {
542           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543             { ns } \l_tmpb_tl
544         }
545         {ns} {
546           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547             { ns } \l_tmpb_tl
548         }
549         {dependencies} {
550           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
551             { deps } \l_tmpb_tl
552         }
553       }{}{}
554     }{}
555   }
556   \ior_close:N \c__stex_mathhub_manifest_ior
557 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

558 \cs_new_protected:Nn \stex_set_current_repository:n {
559   \stex_require_repository:n { #1 }
560   \prop_set_eq:Nc \l_stex_current_repository_prop {
561     c_stex_mathhub_#1_manifest_prop
562   }
563 }

```

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

`\stex_require_repository:n`

```

564 \cs_new_protected:Nn \stex_require_repository:n {
565   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
566     \stex_debug:nn{mathhub}{Opening~archive:~#1}
567     \__stex_mathhub_do_manifest:n { #1 }
568     \exp_args:Nx \stex_add_to_sms:n {
569       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
570         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
571         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
572         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
573         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
574       }
575     }
576   }
577 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

578 %\prop_new:N \l_stex_current_repository_prop
579
580 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
581 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
582   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
583 } {
584   \__stex_mathhub_parse_manifest:n { main }
585   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
586   \l_tmpa_str
587   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
588   \c_stex_mathhub_main_manifest_prop
589   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
590   \stex_debug:nn{mathhub}{Current~repository:~
591     \prop_item:Nn \l_stex_current_repository_prop {id}
592   }
593 }

```

(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.

```

594 \cs_new_protected:Nn \stex_in_repository:nn {
595   \str_set:Nx \l_tmpa_str { #1 }
596   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
597   \str_if_empty:NTF \l_tmpa_str {
598     \prop_if_exist:NTF \l_stex_current_repository_prop {
599       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
600       \exp_args:Ne \l_tmpa_cs{
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \l_tmpa_cs{}
605     }
606   }{
607     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

608 \stex_require_repository:n \l_tmpa_str
609 \str_set:Nx \l_tmpa_str { #1 }
610 \exp_args:Nne \use:nn {
611   \stex_set_current_repository:n \l_tmpa_str
612   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
613 }{
614   \stex_debug:nn{mathhub}{switching~back~to:~
615     \prop_if_exist:NTF \l_stex_current_repository_prop {
616       \prop_item:Nn \l_stex_current_repository_prop { id }::~
617       \meaning\l_stex_current_repository_prop
618     }{
619       no~repository
620     }
621   }
622   \prop_if_exist:NTF \l_stex_current_repository_prop {
623     \stex_set_current_repository:n {
624       \prop_item:Nn \l_stex_current_repository_prop { id }
625     }
626   }{
627     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
628   }
629 }
630 }
631 }

```

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

```

\inputref
\stex_inputref:nn 632 \newif \ifinputref \inputreffalse
\mhinput\stex_mhinput:nn 633
634 \cs_new_protected:Nn \stex_mhinput:nn {
635   \stex_in_repository:nn {#1} {
636     \ifinputref
637       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
638     \else
639       \inputreftrue
640       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
641       \inputreffalse
642     \fi
643   }
644 }
645 \NewDocumentCommand \mhinput { 0{} m}{
646   \stex_mhinput:nn{ #1 }{ #2 }
647 }
648
649 \cs_new_protected:Nn \stex_inputref:nn {
650   \stex_in_repository:nn {#1} {
651     \bool_lazy_any:nTF {
652       {\rustex_if_p:} {\latexml_if_p:}
653     } {
654       \str_clear:N \l_tmpa_str
655       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
656         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
657       }

```

```

658     \stex_annotate_invisible:nnn{inputref}{
659       \l_tmpa_str / #2
660     }{}
661   }{
662     \begingroup
663       \inputreftrue
664       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
665     \endgroup
666   }
667 }
668 }
669
670 \NewDocumentCommand \inputref { 0{ } m }{
671   \stex_inputref:nn{ #1 }{ #2 }
672 }
673
674 \cs_new_protected:Nn \stex_mhbibresource:nn {
675   \stex_in_repository:nn {#1} {
676     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
677   }
678 }
679 \newcommand\addmhbibresource[2][]{
680   \stex_mhbibresource:nn{ #1 }{ #2 }
681 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page 24.)

`\mhpath`

```

682 \def \mhpath #1 #2 {
683   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
684     \c_stex_mathhub_str /
685     \prop_item:Nn \l_stex_current_repository_prop { id }
686     / source / #2
687   }{
688     \c_stex_mathhub_str / #1 / source / #2
689   }
690 }

```

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

`\libinput`

```

691 \cs_new_protected:Npn \libinput #1 {
692   \prop_if_exist:NF \l_stex_current_repository_prop {
693     \msg_error:nnn{stex}{error/notinarchive}\libinput
694   }
695   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
696     \msg_error:nnn{stex}{error/notinarchive}\libinput
697   }
698   \bool_set_false:N \l_tmpa_bool
699   \tl_clear:N \l_tmpa_tl
700   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
701   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
702   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
703   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

704 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
705 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
706 / meta-inf / lib / #1.tex}{
707 \bool_set_true:N \l_tmpa_bool
708 \tl_put_right:Nx \l_tmpa_tl {
709 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
710 / meta-inf / lib / #1.tex}
711 }
712 }{}
713 }
714 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
715 / \l_tmpa_str / lib / #1.tex
716 }{
717 \bool_set_true:N \l_tmpa_bool
718 \tl_put_right:Nx \l_tmpa_tl {
719 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
720 / \l_tmpa_str / lib / #1.tex}
721 }
722 }{}
723 \bool_if:NF \l_tmpa_bool {
724 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
725 }
726 \l_tmpa_tl
727 }

```

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

`\libusepackage`

```

728 \NewDocumentCommand \libusepackage {0{} m} {
729 \prop_if_exist:NF \l_stex_current_repository_prop {
730 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
731 }
732 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
733 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
734 }
735 \bool_set_false:N \l_libusepackage_bool
736 \tl_clear:N \l_tmpa_tl
737 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
738 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
739 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
740 \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
741 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
742 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
743 / meta-inf / lib / #2.sty}{
744 \bool_set_true:N \l_libusepackage_bool
745 \tl_put_right:Nx \l_tmpa_tl {
746 \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
747 / meta-inf / lib / #2}
748 }
749 }{}
750 }
751 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
752 / \l_tmpa_str / lib / #2.sty
753 }{

```



```

754 \bool_if:NT \l_libusepackage_bool {
755   \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
756 }
757 \bool_set_true:N \l_libusepackage_bool
758 \tl_put_right:Nx \l_tmpa_tl {
759   \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
760     / \l_tmpa_str / lib / #2}
761 }
762 }{}
763 \bool_if:NF \l_libusepackage_bool {
764   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
765 }
766 \l_tmpa_tl
767 }

```

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

```

768
769 \AddToHook{begindocument}{
770 \ltx@ifpackageloaded{graphicx}{
771   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
772   \newcommand\mhgraphics[2][]{%
773     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
774     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
775   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
776 }{}
777 \ltx@ifpackageloaded{listings}{
778   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
779   \newcommand\lstinputmhlsting[2][]{%
780     \def\lst@mhrepos{}\setkeys{lst}{#1}%
781     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
782   \newcommand\clstinputmhlsting[2][]{\begin{center}\lstinputmhlsting[#1]{#2}\end{center}}
783 }{}
784 }
785
786
787 \end{package}

```

Chapter 27

STEX -References Implementation

```
788 <*package>
789
790 %%%%%%%%%% references.dtx %%%%%%%%%%
791
792 %\RequirePackage{hyperref}
793 %\RequirePackage{cleveref}
794 <@@=stex_refs>
795
796 Warnings and error messages
797
798 \iow_new:N \c__stex_refs_refs_iow
799 \AddToHook{begindocument}{
800   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
801 }
802 \AddToHook{enddocument}{
803   \iow_close:N \c__stex_refs_refs_iow
804 }
805
806 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
807
808 \NewDocumentCommand \STEXreftitle { m } {
809   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
810 }
811
```

27.1 Document URIs and URLs

```
809 \seq_new:N \g__stex_refs_all_refs_seq
810
811 \str_new:N \l_stex_current_docns_str
812
813 \cs_new_protected:Nn \stex_get_document_uri: {
814   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
815   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
816   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
817   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
818 }
819
```

```

818 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
819
820 \str_clear:N \l_tmpa_str
821 \prop_if_exist:NT \l_stex_current_repository_prop {
822   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824   }
825 }
826
827 \str_if_empty:NTF \l_tmpa_str {
828   \str_set:Nx \l_stex_current_docns_str {
829     file:/\stex_path_to_string:N \l_tmpa_seq
830   }
831 }{
832   \bool_set_true:N \l_tmpa_bool
833   \bool_while_do:Nn \l_tmpa_bool {
834     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
835     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
836       {source} { \bool_set_false:N \l_tmpa_bool }
837     }{}{
838       \seq_if_empty:NT \l_tmpa_seq {
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843
844   \seq_if_empty:NTF \l_tmpa_seq {
845     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
846   }{
847     \str_set:Nx \l_stex_current_docns_str {
848       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
849     }
850   }
851 }
852 }
853
854 \str_new:N \l_stex_current_docurl_str
855 \cs_new_protected:Nn \stex_get_document_url: {
856   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \str_clear:N \l_tmpa_str
863   \prop_if_exist:NT \l_stex_current_repository_prop {
864     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
865       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
866         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
867       }
868     }
869
870   \str_if_empty:NTF \l_tmpa_str {
871     \str_set:Nx \l_stex_current_docurl_str {

```

```

872     file:/\stex_path_to_string:N \l_tmpa_seq
873   }
874 }{
875   \bool_set_true:N \l_tmpa_bool
876   \bool_while_do:Nn \l_tmpa_bool {
877     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
878     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
879       {source} { \bool_set_false:N \l_tmpa_bool }
880     }{}{
881       \seq_if_empty:NT \l_tmpa_seq {
882         \bool_set_false:N \l_tmpa_bool
883       }
884     }
885   }
886
887   \seq_if_empty:NTF \l_tmpa_seq {
888     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
889   }{
890     \str_set:Nx \l_stex_current_docurl_str {
891       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
892     }
893   }
894 }
895 }

```

27.2 Setting Reference Targets

```

896 \str_const:Nn \c__stex_refs_url_str{URL}
897 \str_const:Nn \c__stex_refs_ref_str{REF}
898 % @currentlabel -> number
899 % @currentlabelname -> title
900 % @currentHref -> name.number <- id of some kind
901 % \theH# -> \arabic{section}
902 % \the# -> number
903 % \hyper@makecurrent{#}
904 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
905   \stex_get_document_uri:
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NT \l_tmpa_str {
908     \int_zero:N \l_tmpa_int
909     \bool_set_true:N \l_tmpa_bool
910     \bool_while_do:Nn \l_tmpa_bool {
911       \cs_if_exist:cTF {
912         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
913       }{
914         \int_incr:N \l_tmpa_int
915       }{
916         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
917         \bool_set_false:N \l_tmpa_bool
918       }
919     }
920   }
921   \str_set:Nx \l_tmpa_str {
922     \l_stex_current_docns_str??\l_tmpa_str

```

```

923 }
924 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
925 \stex_if_smsmode:TF {
926   \stex_get_document_url:
927   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
928   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
929 }{
930   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
931   \exp_args:Nx\label{sref_\l_tmpa_str}
932   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
933   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
934 }
935 }
936 \cs_new_protected:Npn \stexauxadddocref #1 {
937   \str_set:Nx \l_tmpa_str {#1}
938   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
939   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
940 }
941 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
942   \stex_get_document_uri:
943   \stex_if_smsmode:TF {
944     \stex_get_document_url:
945     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
946     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
947   }
948   }{
949     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
950     \exp_args:Nx\label{sref_sym_#1}
951
952     \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
953     \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
954   }
955 }

```

27.3 Using References

```

956 \str_new:N \l__stex_refs_indocument_str
957 \keys_define:nn { stex / sref } {
958   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
959   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
960   pre           .tl_set:N = \l__stex_refs_pre_tl ,
961   post          .tl_set:N = \l__stex_refs_post_tl ,
962   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
963 }
964
965 \bool_new:N \c__stex_refs_hyperref_bool
966 \bool_set_false:N \c__stex_refs_hyperref_bool
967 \AddToHook{begindocument}{
968   \@ifpackageloaded{hyperref}{
969     \bool_set_true:N \c__stex_refs_hyperref_bool
970   }{}
971 }
972
973

```

```

974 \cs_new_protected:Nn \__stex_refs_args:n {
975   \tl_clear:N \l__stex_refs_linktext_tl
976   \tl_clear:N \l__stex_refs_fallback_tl
977   \tl_clear:N \l__stex_refs_pre_tl
978   \tl_clear:N \l__stex_refs_post_tl
979   \str_clear:N \l__stex_refs_repo_str
980   \keys_set:nn { stex / sref } { #1 }
981 }
982
983 \NewDocumentCommand \sref { 0{} m}{
984   \__stex_refs_args:n { #1 }
985   \str_if_empty:NTF \l__stex_refs_indocument_str {
986     \str_set:Nn \l_tmpa_str { #2 }
987     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
988     \tl_set:Nn \l_tmpa_tl {
989       \l__stex_refs_fallback_tl
990     }
991     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
992       \str_set:Nn \l_tmpb_str { ##1 }
993       \str_if_eq:eeT { \l_tmpa_str } {
994         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
995       } {
996         \seq_map_break:n {
997           \tl_set:Nn \l_tmpa_tl {
998             % doc uri in \l_tmpb_str
999             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
1000             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1001               % reference
1002               \cs_if_exist:cTF{autoref}{
1003                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1004               }{
1005                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1006               }
1007             }{
1008               % URL
1009               \if_bool:N \c__stex_refs_hyperref_bool {
1010                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1011               }{
1012                 \l__stex_refs_fallback_tl
1013               }
1014             }
1015           }
1016         }
1017       }
1018     }
1019     \l_tmpa_tl
1020   }{
1021     % TODO
1022   }
1023 }
1024
1025 \NewDocumentCommand \srefsym { 0{} m}{
1026   \stex_get_symbol:n { #2 }
1027   \__stex_refs_args:n { #1 }

```

```

1028 \str_if_empty:NTF \l__stex_refs_indocument_str {
1029   \tl_set:Nn \l_tmpa_tl {
1030     \l__stex_refs_fallback_tl
1031   }
1032   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1033     \tl_set:Nn \l_tmpa_tl {
1034       % doc uri in \l_tmpb_str
1035       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1036       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1037         % reference
1038         \cs_if_exist:cTF{autoref}{
1039           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1040         }{
1041           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1042         }
1043       }{
1044         % URL
1045         \if_bool:N \c__stex_refs_hyperref_bool {
1046           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1047         }{
1048           \l__stex_refs_fallback_tl
1049         }
1050       }
1051     }
1052   }
1053   \l_tmpa_tl
1054 }{
1055   % TODO
1056 }
1057 }
1058
1059 \cs_new_protected:Npn \srefsymuri #1 #2 {
1060   \hyperref[sref_sym_#1]{#2}
1061 }
1062
1063 </package>

```

Chapter 28

STEX -Modules Implementation

```
1064 <*package>
1065
1066 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1067
1068 <@@=stex_modules>
1069
1070 Warnings and error messages
1071 \msg_new:nnn{stex}{error/unknownmodule}{
1072   No~module~#1~found
1073 }
1074 \msg_new:nnn{stex}{error/syntax}{
1075   Syntax~error:~#1
1076 }
1077 \msg_new:nnn{stex}{error/siglanguage}{
1078   Module~#1~declares~signature~#2,~but~does~not~
1079   declare~its~language
1080 }
1081 \msg_new:nnn{stex}{error/conflictingmodules}{
1082   Conflicting~imports~for~module~#1
1083 }
1084
1085 \l_stex_current_module_str The current module:
1086 \str_new:N \l_stex_current_module_str
1087
1088 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1089
1090 \l_stex_all_modules_seq Stores all available modules
1091 \seq_new:N \l_stex_all_modules_seq
1092
1093 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1094
1095 \stex_if_in_module_p:
1096 \stex_if_in_module:TF
1097 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1098   \str_if_empty:NTF \l_stex_current_module_str
1099   \prg_return_false: \prg_return_true:
1100 }
```


(End definition for `\stex_if_in_module:TF`. This function is documented on page 27.)

`\stex_if_module_exists_p:n`
`\stex_if_module_exists:nTF`

```
1089 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1090   \prop_if_exist:cTF { c_stex_module_#1_prop }
1091   \prg_return_true: \prg_return_false:
1092 }
```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 27.)

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1093 \cs_new_protected:Nn \stex_add_to_current_module:n {
1094   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1095 }
1096 \cs_new_protected:Npn \STEXexport {
1097   \begingroup
1098   \newlinechar=-1\relax
1099   \endlinechar=-1\relax
1100   %\catcode'\ = 9\relax
1101   \expandafter\endgroup\STEXexport:n
1102 }
1103 \cs_new_protected:Nn \STEXexport:n {
1104   \ignorespaces #1
1105   \stex_add_to_current_module:n { \ignorespaces #1 }
1106   \stex_smsmode_do:
1107 }
1108 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 27.)

`\stex_add_constant_to_current_module:n`

```
1109 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1110   \str_set:Nx \l_tmpa_str { #1 }
1111   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1112 }
1113
1114 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1115 % \str_set:Nx \l_tmpa_str { #1 }
1116 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1117 %}
```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 27.)

`\stex_collect_imports:n`

```
1118 \cs_new_protected:Nn \stex_collect_imports:n {
1119   \seq_clear:N \l_stex_collect_imports_seq
1120   \__stex_modules_collect_imports:n {#1}
1121 }
1122 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1123   \seq_map_inline:cn {c_stex_module_#1_imports} {
1124     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1125       \__stex_modules_collect_imports:n { ##1 }
1126     }
1127 }
```

```

1127 }
1128 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1129   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1130 }
1131 }

```

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

`\stex_add_import_to_current_module:n`

```

1132 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1133   \str_set:Nx \l_tmpa_str { #1 }
1134   \exp_args:Nno
1135   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1136     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1137   }
1138 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 27.)

`\stex_modules_compute_namespace:nN`

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

```

1139 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \seq_set_eq:NN \l_tmpa_seq #2
1142   % split off file extension
1143   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1144   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1145   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1146   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1147
1148   \bool_set_true:N \l_tmpa_bool
1149   \bool_while_do:Nn \l_tmpa_bool {
1150     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1151     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1152       {source} { \bool_set_false:N \l_tmpa_bool }
1153     }{}{
1154       \seq_if_empty:NT \l_tmpa_seq {
1155         \bool_set_false:N \l_tmpa_bool
1156       }
1157     }
1158   }
1159
1160   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1161   \str_if_empty:NTF \l_stex_modules_subpath_str {
1162     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1163   }{
1164     \str_set:Nx \l_stex_modules_ns_str {
1165       \l_tmpa_str/\l_stex_modules_subpath_str
1166     }
1167   }
1168 }

```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 27.)

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1169 \str_new:N \l_stex_modules_ns_str
1170 \str_new:N \l_stex_modules_subpath_str

(End definition for \l_stex_modules_ns_str and \l_stex_modules_subpath_str. These variables are
documented on page ??.)

```

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

```

1171 \cs_new_protected:Nn \stex_modules_current_namespace: {
1172   \str_clear:N \l_stex_modules_subpath_str
1173   \prop_if_exist:NTF \l_stex_current_repository_prop {
1174     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1175     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1176   }{
1177     % split off file extension
1178     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1179     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1180     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1182     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1183     \str_set:Nx \l_stex_modules_ns_str {
1184       file:/\stex_path_to_string:N \l_tmpa_seq
1185     }
1186   }
1187 }

```

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

28.1 The module environment

module arguments:

```

1188 \keys_define:nn { stex / module } {
1189   title      .tl_set:N      = \smodulename ,
1190   type       .str_set_x:N   = \smodulename ,
1191   id         .str_set_x:N   = \smoduleid ,
1192   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1193   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1194   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1195   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1196   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1197   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1198   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1199 }
1200
1201 \cs_new_protected:Nn \__stex_modules_args:n {
1202   \str_clear:N \smodulename
1203   \str_clear:N \smodulename
1204   \str_clear:N \smoduleid
1205   \str_clear:N \l_stex_module_ns_str
1206   \str_clear:N \l_stex_module_lang_str
1207   \str_clear:N \l_stex_module_sig_str
1208   \str_clear:N \l_stex_module_creators_str

```

```

1209 \str_clear:N \l_stex_module_contributors_str
1210 \str_clear:N \l_stex_module_meta_str
1211 \str_clear:N \l_stex_module_srccite_str
1212 \keys_set:nn { stex / module } { #1 }
1213 }
1214
1215 % module parameters here? In the body?
1216

```

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

```

1217 \cs_new_protected:Nn \stex_module_setup:nn {
1218   \str_set:Nx \l_stex_module_name_str { #2 }
1219   \__stex_modules_args:n { #1 }

   First, we set up the name and namespace of the module.
   Are we in a nested module?

1220   \stex_if_in_module:TF {
1221     % Nested module
1222     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1223       { ns } \l_stex_module_ns_str
1224     \str_set:Nx \l_stex_module_name_str {
1225       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1226         { name } / \l_stex_module_name_str
1227     }
1228   }{
1229     % not nested:
1230     \str_if_empty:NT \l_stex_module_ns_str {
1231       \stex_modules_current_namespace:
1232       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1233       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1234         / { \l_stex_module_ns_str }
1235       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1236       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1237         \str_set:Nx \l_stex_module_ns_str {
1238           \stex_path_to_string:N \l_tmpa_seq
1239         }
1240       }
1241     }
1242   }

```

Next, we determine the language of the module:

```

1243 \str_if_empty:NT \l_stex_module_lang_str {
1244   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1245   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1246   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1247   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1248   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1249     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1250       inferred~from~file~name}
1251     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1252   }
1253 }
1254
1255 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1256 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1257 \l_tmpa_str {
1258   \ltx@ifpackageloaded{babel}{
1259     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1260   }{}
1261 } {
1262   \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1263 }
1264 }}

```

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

```

1265 \str_if_empty:NTF \l_stex_module_sig_str {
1266   \exp_args:Nnx \prop_gset_from_keyval:cn {
1267     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1268   } {
1269     name      = \l_stex_module_name_str ,
1270     ns        = \l_stex_module_ns_str ,
1271     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1272     lang      = \l_stex_module_lang_str ,
1273     sig       = \l_stex_module_sig_str ,
1274     meta      = \l_stex_module_meta_str
1275   }
1276   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1277   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1278   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1279   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1280   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1281 \str_if_empty:NT \l_stex_module_meta_str {
1282   \str_set:Nx \l_stex_module_meta_str {
1283     \c_stex_metatheory_ns_str ? Metatheory
1284   }
1285 }
1286 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1287   \bool_set_true:N \l_stex_in_meta_bool
1288   \exp_args:Nx \stex_add_to_current_module:n {
1289     \bool_set_true:N \l_stex_in_meta_bool
1290     \stex_activate_module:n {\l_stex_module_meta_str}
1291     \bool_set_false:N \l_stex_in_meta_bool
1292   }
1293   \stex_activate_module:n {\l_stex_module_meta_str}
1294   \bool_set_false:N \l_stex_in_meta_bool
1295 }
1296 }{
1297   \str_if_empty:NT \l_stex_module_lang_str {
1298     \msg_error:nnxx{stex}{error/siglanguage}{
1299       \l_stex_module_ns_str?\l_stex_module_name_str
1300     }\l_stex_module_sig_str}
1301   }
1302
1303   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1304   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1305 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1306 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1307 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1308 \str_set:Nx \l_tmpa_str {
1309   \stex_path_to_string:N \l_tmpa_seq /
1310   \l_tmpa_str . \l_stex_module_sig_str .tex
1311 }
1312 \IfFileExists \l_tmpa_str {
1313   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1314     \str_clear:N \l_stex_current_module_str
1315     \seq_clear:N \l_stex_all_modules_seq
1316     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1317   }
1318 }{
1319   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1320 }
1321 \stex_if_smsmode:F {
1322   \stex_activate_module:n {
1323     \l_stex_module_ns_str ? \l_stex_module_name_str
1324   }
1325 }
1326 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1327 }
1328 }

```

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

module The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1329 \int_new:N \l_stex_module_group_depth_int
1330 \cs_new_protected:Nn \__stex_modules_begin_module: {
1331   \stex_reactivate_macro:N \STEXexport
1332   \stex_reactivate_macro:N \importmodule
1333   \stex_reactivate_macro:N \symdecl
1334   \stex_reactivate_macro:N \notation
1335   \stex_reactivate_macro:N \symdef
1336
1337   \stex_debug:nn{modules}{
1338     New~module:\\
1339     Namespace:~\l_stex_module_ns_str\\
1340     Name:~\l_stex_module_name_str\\
1341     Language:~\l_stex_module_lang_str\\
1342     Signature:~\l_stex_module_sig_str\\
1343     Metatheory:~\l_stex_module_meta_str\\
1344     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1345   }
1346
1347   \seq_put_right:Nx \l_stex_all_modules_seq {
1348     \l_stex_module_ns_str ? \l_stex_module_name_str
1349   }
1350
1351   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1352   % { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1353
1354
1355 \stex_if_smsmode:F{
1356   \begin{stex_annotate_env} {theory} {
1357     \l_stex_module_ns_str ? \l_stex_module_name_str
1358   }
1359
1360   \stex_annotate_invisible:nnn{header}{} {
1361     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1362     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1363     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1364       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1365     }
1366     \str_if_empty:NF \smoduletype {
1367       \stex_annotate:nnn{type}{\smoduletype}{}
1368     }
1369   }
1370 }
1371 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1372 % TODO: Inherit metatheory for nested modules?
1373 }
1374 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for _stex_modules_begin_module:.)

```

\_stex_modules_end_module: implements \end{module}

1375 \cs_new_protected:Nn \_stex_modules_end_module: {
1376   % \str_set:Nx \l_tmpa_str {
1377   %   c_stex_module_
1378   %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1379   %   \prop_item:Nn \l_stex_current_module_prop { name }
1380   %   _prop
1381   % }
1382   %^^A \prop_new:c { \l_tmpa_str }
1383   % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1384   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1385   }

```

(End definition for _stex_modules_end_module:.)

smodule The core environment, with no header

```

1386 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1387 \NewDocumentEnvironment { smodule } { 0{} m } {
1388   \stex_module_setup:nn{#1}{#2}
1389   \par
1390   \stex_if_smsmode:F{
1391     \tl_clear:N \l_tmpa_tl
1392     \clist_map_inline:Nn \smoduletype {
1393       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1394         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1395       }
1396     }
1397     \tl_if_empty:NTF \l_tmpa_tl {
1398       \_stex_modules_smodule_start:

```

```

1399     }{
1400         \l_tmpa_tl
1401     }
1402 }
1403 \__stex_modules_begin_module:
1404 \stex_ref_new_doc_target:n \smoduleid
1405 \stex_smsmode_do:
1406 } {
1407 \__stex_modules_end_module:
1408 \stex_if_smsmode:TF {
1409 %     \exp_args:Nx \stex_add_to_sms:n {
1410 %         \prop_gset_from_keyval:cn {
1411 %             c_stex_module_
1412 %             \prop_item:Nn \l_stex_current_module_prop { ns } ?
1413 %             \prop_item:Nn \l_stex_current_module_prop { name }
1414 %             _prop
1415 %         } {
1416 %             name      = \prop_item:cn { \l_tmpa_str } { name } ,
1417 %             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1418 %             file      = \prop_item:cn { \l_tmpa_str } { file } ,
1419 %             lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1420 %             sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1421 %             meta      = \prop_item:cn { \l_tmpa_str } { meta }
1422 %         }
1423 %     }
1424 }{
1425     \end{stex_annotate_env}
1426     \clist_set:No \l_tmpa_clist \smoduletype
1427     \tl_clear:N \l_tmpa_tl
1428     \clist_map_inline:Nn \l_tmpa_clist {
1429         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1430             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1431         }
1432     }
1433     \tl_if_empty:NTF \l_tmpa_tl {
1434         \__stex_modules_smodule_end:
1435     }{
1436         \l_tmpa_tl
1437     }
1438 }
1439 }
1440
1441 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1442 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1443
1444 \newcommand\stexpatchmodule[3] [] {
1445     \str_set:Nx \l_tmpa_str{ #1 }
1446     \str_if_empty:NTF \l_tmpa_str {
1447         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1448         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1449     }{
1450         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1451         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1452     }

```



```

1453 }
1454

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1455 \NewDocumentCommand \STEXModule { m } {
1456   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1457   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1458   \tl_set:Nn \l_tmpa_tl {
1459     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1460   }
1461   \seq_map_inline:Nn \l_stex_all_modules_seq {
1462     \str_set:Nn \l_tmpb_str { ##1 }
1463     \str_if_eq:eeT { \l_tmpa_str } {
1464       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1465     } {
1466       \seq_map_break:n {
1467         \tl_set:Nn \l_tmpa_tl {
1468           \stex_invoke_module:n { ##1 }
1469         }
1470       }
1471     }
1472   }
1473   \l_tmpa_tl
1474 }
1475
1476 \cs_new_protected:Nn \stex_invoke_module:n {
1477   \stex_debug:nn{modules}{Invoking~module~#1}
1478   \peek_charcode_remove:NTF ! {
1479     \__stex_modules_invoke_uri:nN { #1 }
1480   } {
1481     \peek_charcode_remove:NTF ? {
1482       \__stex_modules_invoke_symbol:nn { #1 }
1483     } {
1484       \msg_error:nnx{stex}{error/syntax}{
1485         ?~or~!~expected~after~
1486         \c_backslash_str STEXModule{#1}
1487       }
1488     }
1489   }
1490 }
1491
1492 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1493   \str_set:Nn #2 { #1 }
1494 }
1495
1496 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1497   \stex_invoke_symbol:n{#1?#2}
1498 }

```

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

`\stex_activate_module:n`

```
1499 \bool_new:N \l_stex_in_meta_bool
1500 \bool_set_false:N \l_stex_in_meta_bool
1501 \cs_new_protected:Nn \stex_activate_module:n {
1502   \stex_debug:nn{modules}{Activating~module~#1}
1503   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1504     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1505   }
1506   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1507     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1508     \use:c{ c_stex_module_#1_code }
1509   }
1510 }
```

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

```
1511 \</package>
```

Chapter 29

STEX -Module Inheritance Implementation

```
1512 <*package>
1513
1514 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1515
```

29.1 SMS Mode

```
1516 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1517 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1518 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1519 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1520
1521 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1522   \makeatletter
1523   \makeatother
1524   \ExplSyntaxOn
1525   \ExplSyntaxOff
1526   \rustexBREAK
1527 }
1528
1529 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1530   \symdef
1531   \importmodule
1532   \notation
1533   \symdecl
1534   \STEXexport
1535   \inlineass
1536   \inlinedef
1537   \inlineex
1538   \endinput
1539 }
```

```

1540
1541 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1542   \tl_to_str:n {
1543     smodule,
1544     copymodule,
1545     interpretmodule
1546     sdefinition,
1547     sexample,
1548     sassertion,
1549     sparagraph
1550   }
1551 }

```

(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`

```

1552 \bool_new:N \g__stex_smsmode_bool
1553 \bool_set_false:N \g__stex_smsmode_bool
1554 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1555   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1556 }

```

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

`\stex_in_smsmode:nn`

```

1557 \cs_new_protected:Nn \stex_in_smsmode:nn {
1558   \vbox_set:Nn \l_tmpa_box {
1559     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1560     \bool_gset_true:N \g__stex_smsmode_bool
1561     #2
1562     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1563   }
1564   \box_clear:N \l_tmpa_box
1565 }
1566
1567 \quark_new:N \q__stex_smsmode_break
1568
1569 %\ior_new:N \c__stex_smsmode_ior
1570 %\tl_new:N \l__stex_smsmode_filecontent_tl
1571 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1572   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1573   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1574   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1575     %   \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1576   % }
1577   % \ior_close:N \c__stex_smsmode_ior
1578   \stex_filestack_push:n{#1}
1579   \stex_in_smsmode:nn{#1} {
1580     #2
1581     \everyeof{\q__stex_smsmode_break\noexpand}
1582     \expandafter\expandafter\expandafter
1583     \stex_smsmode_do:
1584     \csname @ @ input\endcsname "#1"\relax
1585     %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl

```

```

1586 }
1587 \stex_filestack_pop:
1588 }

```

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

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

```

1589 \cs_new_protected:Npn \stex_smsmode_do: {
1590   \stex_if_smsmode:T {
1591     \__stex_smsmode_do:w
1592   }
1593 }
1594 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1595   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1596     \expandafter\if\expandafter\relax\noexpand#1
1597     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1598   } \else\expandafter\__stex_smsmode_do:w\fi
1599 }{
1600   \__stex_smsmode_do:w %#1
1601 }
1602 }
1603 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1604   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1605     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1606       #1\__stex_smsmode_do:w
1607     }{
1608       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1609         #1
1610       }{
1611         \cs_if_eq:NNTF \begin #1 {
1612           \__stex_smsmode_check_begin:n
1613         }{
1614           \cs_if_eq:NNTF \end #1 {
1615             \__stex_smsmode_check_end:n
1616           }{
1617             \__stex_smsmode_do:w
1618           }
1619         }
1620       }
1621     }
1622   }
1623 }
1624
1625 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1626   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1627     \begin{#1}
1628   }{
1629     \__stex_smsmode_do:w
1630   }
1631 }
1632 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1633   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1634     \end{#1}\__stex_smsmode_do:w

```

```

1635   }{
1636     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1637   }
1638 }

```

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

29.2 Inheritance

```

1639 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1640 \cs_new_protected:Nn \stex_import_module_uri:nn {
1641   \str_set:Nx \l_stex_import_archive_str { #1 }
1642   \str_set:Nn \l_stex_import_path_str { #2 }
1643
1644   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1645   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1646   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1647
1648   \stex_modules_current_namespace:
1649   \bool_lazy_all:nTF {
1650     {\str_if_empty_p:N \l_stex_import_archive_str}
1651     {\str_if_empty_p:N \l_stex_import_path_str}
1652     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1653   }{
1654     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1655     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1656   }{
1657     \str_if_empty:NT \l_stex_import_archive_str {
1658       \prop_if_exist:NT \l_stex_current_repository_prop {
1659         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1660       }
1661     }
1662     \str_if_empty:NTF \l_stex_import_archive_str {
1663       \str_if_empty:NF \l_stex_import_path_str {
1664         \str_set:Nx \l_stex_import_ns_str {
1665           \l_stex_module_ns_str / \l_stex_import_path_str
1666         }
1667       }
1668     }{
1669       \stex_require_repository:n \l_stex_import_archive_str
1670       \prop_get:cnN { c_stex_mathhub\l_stex_import_archive_str_manifest_prop } { ns }
1671       \l_stex_import_ns_str
1672       \str_if_empty:NF \l_stex_import_path_str {
1673         \str_set:Nx \l_stex_import_ns_str {
1674           \l_stex_import_ns_str / \l_stex_import_path_str
1675         }
1676       }
1677     }
1678   }
1679 }

```

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

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.

```

\l_stex_import_archive_str 1680 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1681 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1682 \str_new:N \l_stex_import_path_str
1683 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1684 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1685   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1686
1687     % archive
1688     \str_set:Nx \l_tmpa_str { #2 }
1689     \str_if_empty:NTF \l_tmpa_str {
1690       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1691     } {
1692       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1693       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1694       \seq_put_right:Nn \l_tmpa_seq { source }
1695     }
1696
1697     % path
1698     \str_set:Nx \l_tmpb_str { #3 }
1699     \str_if_empty:NTF \l_tmpb_str {
1700       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1701
1702       \ltx@ifpackageloaded{babel} {
1703         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1704           { \language } \l_tmpb_str {
1705           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1706         }
1707       } {
1708         \str_clear:N \l_tmpb_str
1709       }
1710
1711       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1712       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1713         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1714       }{
1715         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1716         \IfFileExists{ \l_tmpa_str.tex }{
1717           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1718         }{
1719           % try english as default
1720           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1721           \IfFileExists{ \l_tmpa_str.en.tex }{
1722             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1723           }{
1724             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1725           }
1726         }
1727       }
1728

```

```

1729 } {
1730   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1731   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1732
1733   \ltx@ifpackageloaded{babel} {
1734     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1735       { \language } \l_tmpb_str {
1736       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1737     }
1738   } {
1739     \str_clear:N \l_tmpb_str
1740   }
1741
1742   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1743
1744   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1745   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1746     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1747   }{
1748     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1749     \IfFileExists{ \l_tmpa_str/#4.tex }{
1750       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1751     }{
1752       % try english as default
1753       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1754       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1755         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1756       }{
1757         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1758         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1759           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1760         }{
1761           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1762           \IfFileExists{ \l_tmpa_str.tex }{
1763             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1764           }{
1765             % try english as default
1766             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1767             \IfFileExists{ \l_tmpa_str.en.tex }{
1768               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1769             }{
1770               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1771             }
1772           }
1773         }
1774       }
1775     }
1776   }
1777 }
1778
1779 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1780   \seq_clear:N \l_stex_all_modules_seq
1781   \str_clear:N \l_stex_current_module_str
1782   \str_set:Nx \l_tmpb_str { #2 }

```



```

1783     \str_if_empty:NF \l_tmpb_str {
1784       \stex_set_current_repository:n { #2 }
1785     }
1786     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1787   }
1788
1789   \stex_if_module_exists:nF { #1 ? #4 } {
1790     \msg_error:nnx{stex}{error/unknownmodule}{
1791       #1?#4~(in~file~\g__stex_importmodule_file_str)
1792     }
1793   }
1794 }
1795 \stex_activate_module:n { #1 ? #4 }
1796 }

```

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

`\importmodule`

```

1797 \NewDocumentCommand \importmodule { 0{} m } {
1798   \stex_import_module_uri:nn { #1 } { #2 }
1799   \stex_debug:nn{modules}{Importing~module:~
1800     \l_stex_import_ns_str ? \l_stex_import_name_str
1801   }
1802   \stex_if_smsmode:F {
1803     \stex_import_require_module:nnnn
1804     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1805     { \l_stex_import_path_str } { \l_stex_import_name_str }
1806     \stex_annotate_invisible:nnn
1807     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1808   }
1809   \exp_args:Nx \stex_add_to_current_module:n {
1810     \stex_import_require_module:nnnn
1811     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1812     { \l_stex_import_path_str } { \l_stex_import_name_str }
1813   }
1814   \exp_args:Nx \stex_add_import_to_current_module:n {
1815     \l_stex_import_ns_str ? \l_stex_import_name_str
1816   }
1817   \stex_smsmode_do:
1818 }
1819 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1820 \NewDocumentCommand \usemodule { 0{} m } {
1821   \stex_if_smsmode:F {
1822     \stex_import_module_uri:nn { #1 } { #2 }
1823     \stex_import_require_module:nnnn
1824     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1825     { \l_stex_import_path_str } { \l_stex_import_name_str }
1826     \stex_annotate_invisible:nnn
1827     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1828   }
1829   \stex_smsmode_do:

```

1830 }

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

1831 </package>

Chapter 30

STEX -Symbols Implementation

```
1832 <*package>
1833
1834 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1835
```

Warnings and error messages

```
1836
```

30.1 Symbol Declarations

```
1837 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1838 \seq_new:N \l_stex_all_symbols_seq
```

(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)

`\STEXsymbol`

```
1839 \NewDocumentCommand \STEXsymbol { m } {
1840   \stex_get_symbol:n { #1 }
1841   \exp_args:No
1842   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1843 }
```

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

symdecl arguments:

```
1844 \keys_define:nn { stex / symdecl } {
1845   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1846   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1847   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1848   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1849   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1850   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1851   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1852   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1853 }
```

```

1854
1855 \bool_new:N \l_stex_symdecl_make_macro_bool
1856
1857 \cs_new_protected:Nn \__stex_symdecl_args:n {
1858   \str_clear:N \l_stex_symdecl_name_str
1859   \str_clear:N \l_stex_symdecl_args_str
1860   \bool_set_false:N \l_stex_symdecl_local_bool
1861   \tl_clear:N \l_stex_symdecl_type_tl
1862   \tl_clear:N \l_stex_symdecl_definiens_tl
1863
1864   \keys_set:nn { stex / symdecl } { #1 }
1865 }

```

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

```

1866
1867 \NewDocumentCommand \symdecl { s O{} m } {
1868   \__stex_symdecl_args:n { #2 }
1869   \IfBooleanTF #1 {
1870     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1871   } {
1872     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1873   }
1874   \stex_symdecl_do:n { #3 }
1875   \stex_smsmode_do:
1876 }
1877 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1878 \cs_new_protected:Nn \stex_symdecl_do:n {
1879   \stex_if_in_module:F {
1880     % TODO throw error? some default namespace?
1881   }
1882
1883   \str_if_empty:NT \l_stex_symdecl_name_str {
1884     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1885   }
1886
1887   \prop_if_exist:cT { l_stex_symdecl_
1888     \l_stex_current_module_str ?
1889     \l_stex_symdecl_name_str
1890   }_prop
1891   }{
1892     % TODO throw error (beware of circular dependencies)
1893   }
1894
1895   \prop_clear:N \l_tmpa_prop
1896   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1897   \seq_clear:N \l_tmpa_seq
1898   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1899   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1900

```

```

1901 \exp_args:No \stex_add_constant_to_current_module:n {
1902   \l_stex_symdecl_name_str
1903 }
1904
1905 % arity/args
1906 \int_zero:N \l_tmpb_int
1907
1908 \bool_set_true:N \l_tmpa_bool
1909 \str_map_inline:Nn \l_stex_symdecl_args_str {
1910   \token_case_meaning:NnF ##1 {
1911     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1912     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1913     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1914     {\tl_to_str:n a} {
1915       \bool_set_false:N \l_tmpa_bool
1916       \int_incr:N \l_tmpb_int
1917     }
1918     {\tl_to_str:n B} {
1919       \bool_set_false:N \l_tmpa_bool
1920       \int_incr:N \l_tmpb_int
1921     }
1922   }{
1923     \msg_set:nnn{stex}{error/wrongargs}{
1924       args~value~in~symbol~declaration~for~
1925       \l_stex_current_module_str ?
1926       \l_stex_symdecl_name_str ~
1927       needs~to~be~
1928       i,~a,~b~or~B,~but~##1~given
1929     }
1930     \msg_error:nn{stex}{error/wrongargs}
1931   }
1932 }
1933 \bool_if:NTF \l_tmpa_bool {
1934   % possibly numeric
1935   \str_if_empty:NTF \l_stex_symdecl_args_str {
1936     \prop_put:Nnn \l_tmpa_prop { args } {}
1937     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1938   }{
1939     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1940     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1941     \str_clear:N \l_tmpa_str
1942     \int_step_inline:nn \l_tmpa_int {
1943       \str_put_right:Nn \l_tmpa_str i
1944     }
1945     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1946   }
1947 } {
1948   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1949   \prop_put:Nnx \l_tmpa_prop { arity }
1950     { \str_count:N \l_stex_symdecl_args_str }
1951 }
1952 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1953
1954

```

```

1955 % semantic macro
1956
1957 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1958   \exp_args:Nx \stex_do_aftergroup:n {
1959     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1960       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1961     }}
1962   }
1963
1964   \bool_if:NF \l_stex_symdecl_local_bool {
1965     \exp_args:Nx \stex_add_to_current_module:n {
1966       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1967         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1968       } }
1969     }
1970   }
1971 }
1972
1973 % add to all symbols
1974
1975 \bool_if:NF \l_stex_symdecl_local_bool {
1976   \exp_args:Nx \stex_add_to_current_module:n {
1977     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1978       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1979     }
1980   }
1981 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1982 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1983 %   }
1984 }
1985
1986 \stex_debug:nn{symbols}{New~symbol:~
1987   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1988   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1989   Args:~\prop_item:Nn \l_tmpa_prop { args }
1990 }
1991
1992 % circular dependencies require this:
1993
1994 \prop_if_exist:cF {
1995   l_stex_symdecl_
1996   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1997   _prop
1998 } {
1999   \prop_set_eq:cN {
2000     l_stex_symdecl_
2001     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002     _prop
2003   } \l_tmpa_prop
2004 }
2005
2006 \seq_clear:c {
2007   l_stex_symdecl_
2008   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2009     _notations
2010 }
2011
2012 \bool_if:NF \l_stex_symdecl_local_bool {
2013   \exp_args:Nx
2014   \stex_add_to_current_module:n {
2015     \seq_clear:c {
2016       l_stex_symdecl_
2017       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2018       _notations
2019     }
2020     \prop_set_from_keyval:cn {
2021       l_stex_symdecl_
2022       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2023       _prop
2024     } {
2025       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2026       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2027       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2028       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2029       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2030       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2031     }
2032   }
2033 }
2034
2035 \stex_if_smsmode:TF {
2036   \bool_if:NF \l_stex_symdecl_local_bool {
2037     % \exp_args:Nx \stex_add_to_sms:n {
2038     %   \prop_set_from_keyval:cn {
2039     %     l_stex_symdecl_
2040     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2041     %     _prop
2042     %   } {
2043     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2044     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2045     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2046     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2047     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2048     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2049     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2050     %   }
2051     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2052     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2053     %   }
2054     % }
2055   }
2056 }{
2057   \exp_args:Nx \stex_do_aftergroup:n {
2058     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2059       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2060     }
2061   }
2062   \stex_if_do_html:T {

```

```

2063 \stex_annotate_invisible:nnn {symdecl} {
2064   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2065 } {
2066   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2067   \stex_annotate_invisible:nnn{args}{}{
2068     \prop_item:Nn \l_tmpa_prop { args }
2069   }
2070   \stex_annotate_invisible:nnn{macroname}{#1}{}
2071   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2072     \stex_annotate_invisible:nnn{definiens}{}
2073     { $\l_stex_symdecl_definiens_tl$ }
2074   }
2075 }
2076 }
2077 }
2078 }

```

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

`\stex_get_symbol:n`

```

2079 \str_new:N \l_stex_get_symbol_uri_str
2080
2081 \cs_new_protected:Nn \stex_get_symbol:n {
2082   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2083     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2084   }{
2085     % argument is a string
2086     % is it a command name?
2087     \cs_if_exist:cTF { #1 }{
2088       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2089       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2090       \str_if_empty:NNTF \l_tmpa_str {
2091         \exp_args:Nx \cs_if_eq:NNTF {
2092           \tl_head:N \l_tmpa_tl
2093         } \stex_invoke_symbol:n {
2094           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2095         }{
2096           \__stex_symdecl_get_symbol_from_string:n { #1 }
2097         }
2098       } {
2099         \__stex_symdecl_get_symbol_from_string:n { #1 }
2100       }
2101     }{
2102       % argument is not a command name
2103       \__stex_symdecl_get_symbol_from_string:n { #1 }
2104       % \l_stex_all_symbols_seq
2105     }
2106   }
2107 }
2108
2109 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2110   \str_set:Nn \l_tmpa_str { #1 }
2111   \bool_set_false:N \l_tmpa_bool
2112   \stex_if_in_module:T {

```



```

2113 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2114 \bool_set_true:N \l_tmpa_bool
2115 \str_set:Nx \l_stex_get_symbol_uri_str {
2116 \l_stex_current_module_str ? #1
2117 }
2118 }
2119 }
2120 \bool_if:NF \l_tmpa_bool {
2121 \tl_set:Nn \l_tmpa_tl {
2122 \msg_set:nnn{stex}{error/unknownsymbol}{
2123 No~symbol~#1~found!
2124 }
2125 \msg_error:nn{stex}{error/unknownsymbol}
2126 }
2127 \str_set:Nn \l_tmpa_str { #1 }
2128 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2129 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2130 \str_set:Nn \l_tmpb_str { ##1 }
2131 \str_if_eq:eeT { \l_tmpa_str } {
2132 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2133 } {
2134 \seq_map_break:n {
2135 \tl_set:Nn \l_tmpa_tl {
2136 \str_set:Nn \l_stex_get_symbol_uri_str {
2137 ##1
2138 }
2139 }
2140 }
2141 }
2142 }
2143 \l_tmpa_tl
2144 }
2145 }
2146
2147 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2148 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2149 { \tl_tail:N \l_tmpa_tl }
2150 \tl_if_single:NTF \l_tmpa_tl {
2151 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2152 \exp_after:wN \str_set:Nn \exp_after:wN
2153 \l_stex_get_symbol_uri_str \l_tmpa_tl
2154 }{
2155 % TODO
2156 % tail is not a single group
2157 }
2158 }{
2159 % TODO
2160 % tail is not a single group
2161 }
2162 }

```

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

30.2 Notations

```

2163 <@@=stex_notation>

      notation arguments:
2164 \keys_define:nn { stex / notation } {
2165   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2166   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2167   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2168   op        .tl_set:N   = \l__stex_notation_op_tl ,
2169   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2170   primary   .default:n  = {true} ,
2171   unknown   .code:n     = \str_set:Nx
2172             \l__stex_notation_variant_str \l_keys_key_str
2173 }
2174
2175 \cs_new_protected:Nn \stex_notation_args:n {
2176   \str_clear:N \l__stex_notation_lang_str
2177   \str_clear:N \l__stex_notation_variant_str
2178   \str_clear:N \l__stex_notation_prec_str
2179   \tl_clear:N \l__stex_notation_op_tl
2180   \bool_set_false:N \l__stex_notation_primary_bool
2181
2182   \keys_set:nn { stex / notation } { #1 }
2183 }

```

\notation

```

2184 \NewDocumentCommand \notation { 0{ } m } {
2185   \stex_notation_args:n { #1 }
2186   \tl_clear:N \l_stex_symdecl_definiens_tl
2187   \stex_get_symbol:n { #2 }
2188   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2189 }
2190 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nn

```

2191 \cs_new_protected:Nn \stex_notation_do:nn {
2192   \let\l_stex_current_symbol_str\relax
2193   \prop_set_eq:Nc \l_tmpa_prop {
2194     \l_stex_symdecl_ #1 _prop
2195   }
2196
2197   \prop_clear:N \l_tmpb_prop
2198   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2199   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2200   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2201
2202   % precedences
2203   \seq_clear:N \l_tmpb_seq
2204   \exp_args:NNno
2205   \str_if_empty:NTF \l__stex_notation_prec_str {
2206     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2207     \int_compare:nNnTF \l_tmpa_str = 0 {

```

```

2208     \exp_args:NNnx
2209     \prop_put:Nno \l_tmpb_prop { opprec }
2210     { \neginfprec }
2211   }{
2212     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2213   }
2214 } {
2215   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2216     \exp_args:NNnx
2217     \prop_put:Nno \l_tmpb_prop { opprec }
2218     { \neginfprec }
2219     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2220     \int_step_inline:nn { \l_tmpa_str } {
2221       \exp_args:NNx
2222       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2223     }
2224   }{
2225     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2226     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2227       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2228       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2229         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2230         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2231         \seq_map_inline:Nn \l_tmpa_seq {
2232           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2233         }
2234       }
2235       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2236     }{
2237       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2238       \int_compare:nNnTF \l_tmpa_str = 0 {
2239         \exp_args:NNnx
2240         \prop_put:Nno \l_tmpb_prop { opprec }
2241         { \infprec }
2242       }{
2243         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2244       }
2245     }
2246   }
2247 }
2248
2249 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2250 \int_step_inline:nn { \l_tmpa_str } {
2251   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2252     \exp_args:NNx
2253     \seq_put_right:Nn \l_tmpb_seq {
2254       \prop_item:Nn \l_tmpb_prop { opprec }
2255     }
2256   }
2257 }
2258
2259 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2260 \tl_clear:N \l_tmpa_tl
2261

```

```

2262 \int_compare:nNnTF \l_tmpa_str = 0 {
2263   \exp_args:NNe
2264   \cs_set:Npn \l__stex_notation_macrocode_cs {
2265     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2266     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2267     { \prop_item:Nn \l_tmpb_prop { opprec } }
2268     { \exp_not:n { #2 } }
2269   }
2270   \__stex_notation_final:
2271 }{
2272   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2273   \str_if_in:NnTF \l_tmpb_str b {
2274     \exp_args:Nne \use:nn
2275     {
2276       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2277       \cs_set:Npn \l_tmpa_str } { {
2278         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2279         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2280         { \prop_item:Nn \l_tmpb_prop { opprec } }
2281         { \exp_not:n { #2 } }
2282       }}
2283   }{
2284     \str_if_in:NnTF \l_tmpb_str B {
2285       \exp_args:Nne \use:nn
2286       {
2287         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2288         \cs_set:Npn \l_tmpa_str } { {
2289           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2290           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2291           { \prop_item:Nn \l_tmpb_prop { opprec } }
2292           { \exp_not:n { #2 } }
2293         } }
2294     }{
2295       \exp_args:Nne \use:nn
2296       {
2297         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2298         \cs_set:Npn \l_tmpa_str } { {
2299           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2300           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2301           { \prop_item:Nn \l_tmpb_prop { opprec } }
2302           { \exp_not:n { #2 } }
2303         } }
2304     }
2305   }
2306
2307   \int_zero:N \l_tmpa_int
2308   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2309   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2310   \__stex_notation_arguments:
2311 }
2312 }

```

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

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

```

2313 \cs_new_protected:Nn \_stex_notation_arguments: {
2314   \int_incr:N \l_tmpa_int
2315   \str_if_empty:NTF \l_tmpa_str {
2316     \_stex_notation_final:
2317   }{
2318     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2319     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2320     \str_if_eq:VnTF \l_tmpb_str a {
2321       \_stex_notation_argument_assoc:n
2322     }{
2323       \str_if_eq:VnTF \l_tmpb_str B {
2324         \_stex_notation_argument_assoc:n
2325       }{
2326         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2327         \tl_put_right:Nx \l_tmpa_tl {
2328           { \_stex_term_math_arg:nnn
2329             { \int_use:N \l_tmpa_int }
2330             { \l_tmpb_str }
2331             { ####\int_use:N \l_tmpa_int }
2332           }
2333         }
2334         \_stex_notation_arguments:
2335       }
2336     }
2337   }
2338 }
```

(End definition for `_stex_notation_arguments:.`)

`_stex_notation_argument_assoc:n`

```

2339 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2340   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2341   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2342   \tl_put_right:Nx \l_tmpa_tl {
2343     { \_stex_term_math_assoc_arg:nnnn
2344       { \int_use:N \l_tmpa_int }
2345       { \l_tmpb_str }
2346       \exp_args:No \exp_not:n
2347       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2348       { ####\int_use:N \l_tmpa_int }
2349     }
2350   }
2351   \_stex_notation_arguments:
2352 }
```

(End definition for `_stex_notation_argument_assoc:n.`)

`_stex_notation_final:` Called after processing all notation arguments

```

2353 \cs_new_protected:Nn \_stex_notation_final: {
2354   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2355   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2356   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2357   \exp_args:Nne \use:nn
```

```

2358 {
2359 \cs_generate_from_arg_count:cNnn {
2360   stex_notation_ \l_tmpa_str \c_hash_str
2361   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2362   _cs
2363 }
2364 \cs_set:Npn \l_tmpb_str } { {
2365   \exp_after:wN \exp_after:wN \exp_after:wN
2366   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2367   { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2368 } }
2369
2370 \tl_if_empty:NF \l__stex_notation_op_tl {
2371   \cs_set:cpx {
2372     stex_op_notation_ \l_tmpa_str \c_hash_str
2373     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2374     _cs
2375   } {
2376     \_stex_term_oms:nnn {
2377       \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2378       \l__stex_notation_lang_str
2379     }{
2380       \l_tmpa_str
2381     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2382   }
2383 }
2384
2385 \exp_args:Ne
2386 \stex_add_to_current_module:n {
2387   \cs_generate_from_arg_count:cNnn {
2388     stex_notation_ \l_tmpa_str \c_hash_str
2389     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2390     _cs
2391   } \cs_set:Npn {\l_tmpb_str} {
2392     \exp_after:wN \exp_after:wN \exp_after:wN
2393     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2394     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2395   }
2396   \tl_if_empty:NF \l__stex_notation_op_tl {
2397     \cs_set:cpn {
2398       stex_op_notation_ \l_tmpa_str \c_hash_str
2399       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2400       _cs
2401     } {
2402       \_stex_term_oms:nnn {
2403         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2404         \l__stex_notation_lang_str
2405       }{
2406         \l_tmpa_str
2407       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2408     }
2409   }
2410 }
2411

```

```

2412 \seq_put_right:cx {
2413   l_stex_symdecl_
2414   \prop_item:Nn \l_tmpb_prop { symbol }
2415   _notations
2416 } {
2417   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2418 }
2419
2420 \stex_debug:nn{symbols}{
2421   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2422   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2423   Operator~precedence:~
2424   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2425   Argument~precedences:~
2426   \seq_use:Nn \l_tmpa_seq {,~}^^J
2427   Notation: \cs_meaning:c {
2428     stex_notation_ \l_tmpa_str \c_hash_str
2429     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2430     _cs
2431   }
2432 }
2433
2434 \prop_set_eq:cN {
2435   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2436   \c_hash_str \l__stex_notation_lang_str _prop
2437 } \l_tmpb_prop
2438
2439 \exp_args:Ne
2440 \stex_add_to_current_module:n {
2441   \seq_put_right:cn {
2442     l_stex_symdecl_
2443     \prop_item:Nn \l_tmpb_prop { symbol }
2444     _notations
2445   } {
2446     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2447   }
2448   \prop_set_from_keyval:cn {
2449     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2450     \c_hash_str \l__stex_notation_lang_str _prop
2451   } {
2452     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2453     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2454     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2455     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2456     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2457   }
2458 }
2459
2460 \stex_if_smsmode:TF {
2461 %   \exp_args:Nx \stex_add_to_sms:n {
2462 %     \prop_set_from_keyval:cn {
2463 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2464 %       \c_hash_str \l__stex_notation_lang_str _prop
2465 %     } {

```

```

2466 %      symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2467 %      language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2468 %      variant      = \prop_item:Nn \l_tmpb_prop { variant }      ,
2469 %      opprec       = \prop_item:Nn \l_tmpb_prop { opprec }       ,
2470 %      argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2471 %    }
2472 %  }
2473 }{
2474
2475 % HTML annotations
2476 \stex_if_do_html:T {
2477   \stex_annotate_invisible:nnn { notation }
2478   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2479     \stex_annotate_invisible:nnn { notationfragment }
2480     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2481     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2482     \stex_annotate_invisible:nnn { precedence }
2483     { \prop_item:Nn \l_tmpb_prop { opprec };
2484       \seq_use:Nn \l_tmpa_seq { x }
2485     }{}
2486
2487     \int_zero:N \l_tmpa_int
2488     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2489     \tl_clear:N \l_tmpa_tl
2490     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2491       \int_incr:N \l_tmpa_int
2492       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2493       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2494       \str_if_eq:VnTF \l_tmpb_str a {
2495         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2496           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2497           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2498         } }
2499       }{
2500         \str_if_eq:VnTF \l_tmpb_str B {
2501           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2502             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2503             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2504           } }
2505         }{
2506           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2507             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2508           } }
2509         }
2510       }
2511     }
2512     \stex_annotate_invisible:nnn { notationcomp }{}{
2513       \str_set:Nx \l_stex_current_symbol_str {\prop_item:Nn \l_tmpb_prop { symbol }}
2514       $ \exp_args:Nno \use:nn { \use:c {
2515         stex_notation_ \l_stex_current_symbol_str
2516         \c_hash_str \l__stex_notation_variant_str
2517         \c_hash_str \l__stex_notation_lang_str _cs
2518       } } { \l_tmpa_tl } $
2519     }

```



```

2520     }
2521   }
2522 }
2523 \stex_smsmode_do:
2524 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2525 \keys_define:nn { stex / setnotation } {
2526   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2527   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2528   unknown .code:n = \str_set:Nx
2529     \l__stex_notation_variant_str \l_keys_key_str
2530 }
2531
2532 \cs_new_protected:Nn \_stex_setnotation_args:n {
2533   \str_clear:N \l__stex_notation_lang_str
2534   \str_clear:N \l__stex_notation_variant_str
2535   \keys_set:nn { stex / setnotation } { #1 }
2536 }
2537
2538 \NewDocumentCommand \setnotation {m m} {
2539   \stex_get_symbol:n { #1 }
2540   \_stex_setnotation_args:n { #2 }
2541   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations }
2542     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2543     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2544       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2545     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2546       { \c_hash_str }
2547     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notations
2548       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2549     \exp_args:Nx \stex_add_to_current_module:n {
2550       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2551         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2552       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notation
2553         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2554       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _notati
2555         { \c_hash_str }
2556     }
2557     \stex_debug:nn {notations}{
2558       Setting~default~notation~
2559       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }~for~
2560       \l_stex_get_symbol_uri_str \
2561       \expandafter\meaning\csname
2562       l_stex_symdecl\_l_stex_get_symbol_uri_str _notations\endcsname
2563     }
2564   }{
2565     % todo throw error
2566   }
2567 }
2568

```

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

\symdef

```
2569 \keys_define:nn { stex / symdef } {
2570   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2571   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2572   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2573   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2574   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2575   op        .tl_set:N    = \l__stex_notation_op_tl ,
2576   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2577   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2578   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2579   unknown   .code:n      = \str_set:Nx
2580             \l__stex_notation_variant_str \l_keys_key_str
2581 }
2582
2583 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2584   \str_clear:N \l_stex_symdecl_name_str
2585   \str_clear:N \l_stex_symdecl_args_str
2586   \bool_set_false:N \l_stex_symdecl_local_bool
2587   \tl_clear:N \l_stex_symdecl_type_tl
2588   \tl_clear:N \l_stex_symdecl_definiens_tl
2589   \str_clear:N \l__stex_notation_lang_str
2590   \str_clear:N \l__stex_notation_variant_str
2591   \str_clear:N \l__stex_notation_prec_str
2592   \tl_clear:N \l__stex_notation_op_tl
2593
2594   \keys_set:nn { stex / symdef } { #1 }
2595 }
2596
2597 \NewDocumentCommand \symdef { 0{} m } {
2598   \__stex_notation_symdef_args:n { #1 }
2599   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2600   \stex_symdecl_do:n { #2 }
2601   \exp_args:Nx \stex_notation_do:nn {
2602     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2603   }
2604 }
2605 \stex_deactivate_macro:Nn \symdef {module~environments}
2606
2607 (End definition for \symdef. This function is documented on page 37.)
2608 \endpackage
```

Chapter 31

STEX -Terms Implementation

```
2607 <*package>
2608
2609 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2610
2611 <@@=stex_terms>
2612
2613 Warnings and error messages
2614 \msg_new:nnn{stex}{error/nonotation}{
2615   Symbol~#1~invoked,~but~has~no~notation~#2!
2616 }
2617 \msg_new:nnn{stex}{error/notationarg}{
2618   Error~in~parsing~notation~#1
2619 }
2620 \msg_new:nnn{stex}{error/noop}{
2621   Symbol~#1~has~no~operator~notation~for~notation~#2
2622 }
```

31.1 Symbol Invocations

Arguments:

```
2622 \keys_define:nn { stex / terms } {
2623   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2624   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2625   unknown .code:n = \str_set:Nx
2626     \l__stex_terms_variant_str \l_keys_key_str
2627 }
2628
2629 \cs_new_protected:Nn \__stex_terms_args:n {
2630   \str_clear:N \l__stex_terms_lang_str
2631   \str_clear:N \l__stex_terms_variant_str
2632   \str_clear:N \l__stex_terms_prec_str
2633   \tl_clear:N \l__stex_terms_op_tl
2634 }
2635 \keys_set:nn { stex / terms } { #1 }
```

2636 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2637 \cs_new_protected:Nn \stex_invoke_symbol:n {
2638   \if_mode_math:
2639     \exp_after:wN \__stex_terms_invoke_math:n
2640   \else:
2641     \exp_after:wN \__stex_terms_invoke_text:n
2642   \fi: { #1 }
2643 }
```

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

__stex_terms_invoke_math:n

```
2644 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2645   \peek_charcode_remove:NTF ! {
2646     \peek_charcode:NTF [ {
2647       \__stex_terms_invoke_op:nw { #1 }
2648     }{
2649       \peek_charcode_remove:NTF ! {
2650         \peek_charcode:NTF [ {
2651           \__stex_terms_invoke_op_custom:nw
2652         }{
2653           % TODO throw error
2654         }
2655       }{
2656         \__stex_terms_invoke_op:nw { #1 } []
2657       }
2658     }{
2659       \peek_charcode_remove:NTF * {
2660         \__stex_terms_invoke_text:n { #1 }
2661       }{
2662         \peek_charcode:NTF [ {
2663           \__stex_terms_invoke_math:nw { #1 }
2664         }{
2665           \__stex_terms_invoke_math:nw { #1 } []
2666         }
2667       }
2668     }
2669   }
2670 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2671 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2672   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2673     \stex_highlight_term:nn{#1}{#2}
2674   }
2675 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2676 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2677   \_stex_terms_args:n { #2 }
2678   \cs_if_exist:cTF {
2679     stex_op_notation_ #1 \c_hash_str
2680     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2681   }{
2682     \csname stex_op_notation_ #1 \c_hash_str
2683       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2684     \endcsname
2685   }{
2686     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2687   }
2688 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2689 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2690   \_stex_terms_args:n { #2 }
2691   \seq_if_empty:cTF {
2692     l_stex_symdecl_ #1 _notations
2693   } {
2694     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2695   } {
2696     \seq_if_in:cxTF {
2697       l_stex_symdecl_ #1 _notations
2698     }
2699     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2700       \str_set:Nn \l_stex_current_symbol_str { #1 }
2701       \use:c{
2702         stex_notation_ #1 \c_hash_str
2703         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2704         _cs
2705       }
2706     }{
2707       \str_if_empty:NTF \l__stex_terms_variant_str {
2708         \str_if_empty:NTF \l__stex_terms_lang_str {
2709           \seq_get_left:cN {
2710             l_stex_symdecl_ #1 _notations
2711           } \l_tmpa_str
2712           \str_set:Nn \l_stex_current_symbol_str { #1 }
2713           \use:c{
2714             stex_notation_ #1 \c_hash_str \l_tmpa_str
2715             _cs
2716           }
2717         }{
2718           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2719             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2720           }
2721         }
2722       }{
2723         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2724           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2725     }
2726   }
2727 }
2728 }
2729 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2730 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2731   \peek_charcode_remove:NTF ! {
2732     \stex_term_custom:nn { #1 } { }
2733   }{
2734     \prop_set_eq:Nc \l_tmpa_prop {
2735       l_stex_symdecl_ #1 _prop
2736     }
2737     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2738     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2739   }
2740 }

```

(End definition for `_stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2741 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2742 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2743 \int_new:N \l__stex_terms_downprec
2744 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

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

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2745 \tl_set:Nn \l__stex_terms_left_bracket_str (
2746 \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

```

2747 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2748   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2749     \bool_set_false:N \l__stex_terms_brackets_done_bool
2750     #2
2751   } {
2752     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2753       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2754         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2755         \dobrackets { #2 }
2756       }

```

```

2757     }{ #2 }
2758   }
2759 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2760 \bool_new:N \l__stex_terms_brackets_done_bool
2761 %\RequirePackage{scalerel}
2762 \cs_new_protected:Npn \dobrackets #1 {
2763   %\ThisStyle{\if D\m@switch
2764   %   \exp_args:Nnx \use:nn
2765   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2766   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2767   % \else
2768   %   \exp_args:Nnx \use:nn
2769   %   {
2770   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
2771   %     \int_set:Nn \l__stex_terms_downprec \infprec
2772   %     \l__stex_terms_left_bracket_str
2773   %     #1
2774   %   }
2775   %   {
2776   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
2777   %     \l__stex_terms_right_bracket_str
2778   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2779   %   }
2780   %\fi}
2781 }

```

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

`\withbrackets`

```

2782 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2783   \exp_args:Nnx \use:nn
2784   {
2785     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2786     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2787     #3
2788   }
2789   {
2790     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2791     { \l__stex_terms_left_bracket_str }
2792     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2793     { \l__stex_terms_right_bracket_str }
2794   }
2795 }

```

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

`\STEXinvisible`

```

2796 \cs_new_protected:Npn \STEXinvisible #1 {
2797   \stex_annotate_invisible:n { #1 }
2798 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2799 \cs_new_protected:Nn \_stex_term_oms:nnn {
2800   \stex_annotate:nnn{ OMID }{ #2 }{
2801     \stex_highlight_term:nn { #1 } { #3 }
2802   }
2803 }
2804
2805 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2806   \__stex_terms_maybe_brackets:nn { #3 }{
2807     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2808   }
2809 }

```

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

`_stex_term_math_oma:nnnn`

```

2810 \cs_new_protected:Nn \_stex_term_oma:nnn {
2811   \stex_annotate:nnn{ OMA }{ #2 }{
2812     \stex_highlight_term:nn { #1 } { #3 }
2813   }
2814 }
2815
2816 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2817   \__stex_terms_maybe_brackets:nn { #3 }{
2818     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2819   }
2820 }

```

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

`_stex_term_math_omb:nnnn`

```

2821 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2822   \stex_annotate:nnn{ OMBIND }{ #2 }{
2823     \stex_highlight_term:nn { #1 } { #3 }
2824   }
2825 }
2826
2827 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2828   \__stex_terms_maybe_brackets:nn { #3 }{
2829     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2830   }
2831 }

```

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

`_stex_term_math_arg:nnn`

```

2832 \cs_new_protected:Nn \_stex_term_arg:nn {
2833   \stex_unhighlight_term:n {
2834     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2835   }
2836 }

```



```

2837 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2838   \exp_args:Nnx \use:nn
2839     { \int_set:Nn \l__stex_terms_downprec { #2 }
2840       \stex_term_arg:nn { #1 }{ #3 }
2841     }
2842     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2843   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2844 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2845   \clist_set:Nn \l_tmpa_clist{ #4 }
2846   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2847     \tl_set:Nn \l_tmpa_tl { #4 }
2848   }{
2849     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2850     \clist_reverse:N \l_tmpa_clist
2851     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2852
2853     \clist_map_inline:Nn \l_tmpa_clist {
2854       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2855         \exp_args:Nno
2856           \l_tmpa_cs { ##1 } \l_tmpa_tl
2857       }
2858     }
2859
2860   }
2861   \exp_args:Nnno
2862   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2863 }

```

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

`\stex_term_custom:nn`

```

2864 \cs_new_protected:Nn \stex_term_custom:nn {
2865   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2866   \str_set:Nn \l_tmpa_str { #2 }
2867   \tl_clear:N \l_tmpa_tl
2868   \int_zero:N \l_tmpa_int
2869   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2870   \__stex_terms_custom_loop:
2871 }

```

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

`__stex_terms_custom_loop:`

```

2872 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2873   \bool_set_false:N \l_tmpa_bool
2874   \bool_while_do:nn {
2875     \str_if_eq_p:ee X {
2876       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2877     }
2878   }{
2879     \int_incr:N \l_tmpa_int

```

```

2880 }
2881
2882 \peek_charcode:NTF [ {
2883   % notation/text component
2884   \__stex_terms_custom_component:w
2885 } {
2886   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2887     % all arguments read => finish
2888     \__stex_terms_custom_final:
2889   } {
2890     % arguments missing
2891     \peek_charcode_remove:NTF * {
2892       % invisible, specific argument position or both
2893       \peek_charcode:NTF [ {
2894         % visible specific argument position
2895         \__stex_terms_custom_arg:wn
2896       } {
2897         % invisible
2898         \peek_charcode_remove:NTF * {
2899           % invisible specific argument position
2900           \__stex_terms_custom_arg_inv:wn
2901         } {
2902           % invisible next argument
2903           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2904         }
2905       }
2906     } {
2907       % next normal argument
2908       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2909     }
2910   }
2911 }
2912 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2913 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2914   \bool_set_true:N \l_tmpa_bool
2915   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2916 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2917 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2918   \str_set:Nx \l_tmpb_str {
2919     \str_item:Nn \l_tmpa_str { #1 }
2920   }
2921   \str_case:VnTF \l_tmpb_str {
2922     { X } {
2923       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2924     }
2925     { i } { \__stex_terms_custom_set_X:n { #1 } }
2926     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2927 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2928 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2929 }{}{
2930 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2931 }
2932
2933 \bool_if:nTF \l_tmpa_bool {
2934   \tl_put_right:Nx \l_tmpa_tl {
2935     \stex_annotate_invisible:n {
2936       \stex_term_arg:nn { \int_eval:n { #1 } }
2937       \exp_not:n { { #2 } }
2938     }
2939   }
2940 } {
2941   \tl_put_right:Nx \l_tmpa_tl {
2942     \stex_term_arg:nn { \int_eval:n { #1 } }
2943     \exp_not:n { { #2 } }
2944   }
2945 }
2946
2947 \_stex_terms_custom_loop:
2948 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2949 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2950   \str_set:Nx \l_tmpa_str {
2951     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2952     X
2953     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2954   }
2955 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2956 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2957   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2958   \_stex_terms_custom_loop:
2959 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2960 \cs_new_protected:Nn \_stex_terms_custom_final: {
2961   \int_compare:nNnTF \l_tmpb_int = 0 {
2962     \exp_args:Nnno \stex_term_oms:nnn
2963   }{
2964     \str_if_in:NnTF \l_tmpa_str {b} {
2965       \exp_args:Nnno \stex_term_ombind:nnn
2966     } {
2967       \exp_args:Nnno \stex_term_oma:nnn
2968     }
2969   }

```

```

2970 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2971 }

```

(End definition for `_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2972 \NewDocumentCommand \symref { m m }{
2973   \let\compemph_uri_prev:\compemph@uri
2974   \let\compemph@uri\symrefemph@uri
2975   \STEXsymbol{#1}! [#2]
2976   \let\compemph@uri\compemph_uri_prev:
2977 }
2978
2979 \keys_define:nn { stex / symname } {
2980   post      .str_set_x:N    = \l_stex_symname_post_str
2981 }
2982
2983 \cs_new_protected:Nn \stex_symname_args:n {
2984   \str_clear:N \l_stex_symname_post_str
2985   \keys_set:nn { stex / symname } { #1 }
2986 }
2987
2988 \NewDocumentCommand \symname { 0{} m }{
2989   \stex_symname_args:n { #1 }
2990   \stex_get_symbol:n { #2 }
2991   \str_set:Nx \l_tmpa_str {
2992     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2993   }
2994   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2995
2996   \let\compemph_uri_prev:\compemph@uri
2997   \let\compemph@uri\symrefemph@uri
2998   \exp_args:NNx \use:nn
2999   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3000     \l_tmpa_str \l_stex_symname_post_str
3001   ] }
3002   \let\compemph@uri\compemph_uri_prev:
3003 }

```

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

31.3 Notation Components

```

3004 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3005
3006 \str_new:N \l_stex_current_symbol_str
3007 \cs_new_protected:Nn \stex_highlight_term:nn {
3008   \exp_args:Nnx
3009   \use:nn {
3010     \str_set:Nx \l_stex_current_symbol_str { #1 }
3011     #2
3012   } {

```

```

3013 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3014 { \l_stex_current_symbol_str }
3015 }
3016 }
3017
3018 \cs_new_protected:Nn \stex_unhighlight_term:n {
3019 % \latexml_if:TF {
3020 % #1
3021 % } {
3022 % \rustex_if:TF {
3023 % #1
3024 % } {
3025 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3026 % }
3027 % }
3028 }

```

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

```

\comp
\compemph@uri 3029 \cs_new_protected:Npn \comp #1 {
\compemph 3030 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3031 \rustex_if:TF {
\defemph@uri 3032 \stex_annotate:nnn { comp }{\l_stex_current_symbol_str }{\ #1 }
\symrefemph 3033 }{
\symrefemph@uri 3034 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3035 }
3036 }
3037 }
3038
3039 \cs_new_protected:Npn \compemph@uri #1 #2 {
3040 \compemph{ #1 }
3041 }
3042
3043
3044 \cs_new_protected:Npn \compemph #1 {
3045 #1
3046 }
3047
3048 \cs_new_protected:Npn \defemph@uri #1 #2 {
3049 \defemph{#1}
3050 }
3051
3052 \cs_new_protected:Npn \defemph #1 {
3053 \textbf{#1}
3054 }
3055
3056 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3057 \symrefemph{#1}
3058 }
3059
3060 \cs_new_protected:Npn \symrefemph #1 {
3061 \textbf{#1}
3062 }

```

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

`\ellipses`

```
3063 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 3064 \bool_new:N \l_stex_inarray_bool
\parrayline 3065 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3066 \NewDocumentCommand \parray { m m } {
\parraycell 3067 \begin{group}
3068 \bool_set_true:N \l_stex_inarray_bool
3069 \begin{array}{#1}
3070 #2
3071 \end{array}
3072 \end{group}
3073 }
3074
3075 \NewDocumentCommand \prmatrix { m } {
3076 \begin{group}
3077 \bool_set_true:N \l_stex_inarray_bool
3078 \begin{matrix}
3079 #1
3080 \end{matrix}
3081 \end{group}
3082 }
3083
3084 \def \maybepline {
3085 \bool_if:NT \l_stex_inarray_bool {\hline}
3086 }
3087
3088 \def \parrayline #1 #2 {
3089 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3090 }
3091
3092 \def \pmrow #1 { \parrayline{}{ #1 } }
3093
3094 \def \parraylineh #1 #2 {
3095 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3096 }
3097
3098 \def \parraycell #1 {
3099 #1 \bool_if:NT \l_stex_inarray_bool {&}
3100 }

```

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

```
3101 \end{package}
```

Chapter 32

STEX -Structural Features Implementation

```
3102 <*package>
3103
3104 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3105
3106 <@@=stex_features>
3107
3108     Warnings and error messages
3109 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3110     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3111 }
3112 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3113     Symbol~#1~not~assigned~in~interpretmodule~#2
3114 }
```

32.1 Imports with modification

```
3114 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3115     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3116         \__stex_features_get_symbol_from_cs:n { #1 }
3117     }{
3118         % argument is a string
3119         % is it a command name?
3120         \cs_if_exist:cTF { #1 }{
3121             \cs_set_eq:Nc \l_tmpa_tl { #1 }
3122             \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3123             \str_if_empty:NNTF \l_tmpa_str {
3124                 \exp_args:Nx \cs_if_eq:NNTF {
3125                     \tl_head:N \l_tmpa_tl
3126                 } \stex_invoke_symbol:n {
3127                     \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3128                 }{
3129                     \__stex_features_get_symbol_from_string:n { #1 }
3130                 }
3131             }
3132         }
3133     }
```

```

3130     }
3131   } {
3132     \__stex_features_get_symbol_from_string:n { #1 }
3133   }
3134   ){
3135     % argument is not a command name
3136     \__stex_features_get_symbol_from_string:n { #1 }
3137     % \l_stex_all_symbols_seq
3138   }
3139 }
3140 }
3141
3142 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3143   \str_set:Nn \l_tmpa_str { #1 }
3144   \bool_set_false:N \l_tmpa_bool
3145   \bool_if:NF \l_tmpa_bool {
3146     \tl_set:Nn \l_tmpa_tl {
3147       \msg_set:nnn{stex}{error/unknownsymbol}{
3148         No~symbol~#1~found!
3149       }
3150       \msg_error:nn{stex}{error/unknownsymbol}
3151     }
3152     \str_set:Nn \l_tmpa_str { #1 }
3153     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3154     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3155       \str_set:Nn \l_tmpb_str { ##1 }
3156       \str_if_eq:eeT { \l_tmpa_str } {
3157         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3158       } {
3159         \seq_map_break:n {
3160           \tl_set:Nn \l_tmpa_tl {
3161             \str_set:Nn \l_stex_get_symbol_uri_str {
3162               ##1
3163             }
3164             \__stex_features_get_symbol_check:
3165           }
3166         }
3167       }
3168     }
3169     \l_tmpa_tl
3170   }
3171 }
3172
3173 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3174   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3175   { \tl_tail:N \l_tmpa_tl }
3176   \tl_if_single:NTF \l_tmpa_tl {
3177     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3178       \exp_after:wN \str_set:Nn \exp_after:wN
3179       \l_stex_get_symbol_uri_str \l_tmpa_tl
3180       \__stex_features_get_symbol_check:
3181     }{
3182       % TODO
3183       % tail is not a single group

```



```

3184     }
3185   }{
3186     % TODO
3187     % tail is not a single group
3188   }
3189 }
3190
3191 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3192   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3193   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3194     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3195     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3196     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3197       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3198         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3199       }
3200     }
3201   }{
3202     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3203       \l_stex_current_copymodule_name_str~(inexplicably)
3204     }
3205   }
3206 }
3207
3208 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3209   \stex_import_module_uri:nn { #1 } { #2 }
3210   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3211   \stex_import_require_module:nnnn
3212     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3213     { \l_stex_import_path_str } { \l_stex_import_name_str }
3214   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3215   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3216   \seq_clear:N \l__stex_features_copymodule_fields_seq
3217   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3218     \seq_map_inline:cn {c_stex_module_###1_constants}{
3219       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3220         ###1 ? #####1
3221       }
3222     }
3223   }
3224   \seq_clear:N \l_tmpa_seq
3225   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3226     name      = \l_stex_current_copymodule_name_str ,
3227     module    = \l_stex_current_module_str ,
3228     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3229     includes  = \l_tmpa_seq ,
3230     fields    = \l_tmpa_seq
3231   }
3232   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3233     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3234   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3235     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3236   \stex_if_smsmode:F {
3237     \begin{stex_annotate_env} {#4} {

```

```

3238     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3239   }
3240   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{ }
3241 }
3242 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3243 \bool_set_false:N \l_stex_html_do_output_bool
3244 }
3245 \cs_new_protected:Nn \stex_copymodule_end:n {
3246   \def \l_tmpa_cs ##1 ##2 {#1}
3247   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3248   \tl_clear:N \l_tmpa_tl
3249   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3250   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3251     \seq_map_inline:cn {c_stex_module_##1_constants}{\stex_annotate:nnn{assignment} {##1?###
3252       \l_tmpa_cs{##1}{####1}
3253       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3254         \tl_put_right:Nx \l_tmpa_tl {
3255           \prop_set_from_keyval:cn {
3256             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3257           }{
3258             \exp_after:wN \prop_to_keyval:N \csname
3259               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3260             \endcsname
3261           }
3262           \seq_clear:c {
3263             l_stex_symdecl_
3264             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3265             _notations
3266           }
3267         }
3268         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_nam
3269         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3270         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3271           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?###
3272           \tl_put_right:Nx \l_tmpa_tl {
3273             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3274             \stex_invoke_symbol:n {
3275               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3276             }
3277           }
3278         }
3279       }
3280     }{
3281       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3282       \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3283       \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3284       \tl_put_right:Nx \l_tmpa_tl {
3285         \prop_set_from_keyval:cn {
3286           l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3287         }{
3288           \prop_to_keyval:N \l_tmpa_prop
3289         }
3290       \seq_clear:c {
3291         l_stex_symdecl_

```

```

3292         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3293         _notations
3294     }
3295 }
3296 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1}
3297 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3298     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3299     \tl_put_right:Nx \l_tmpa_tl {
3300         \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3301             \stex_invoke_symbol:n {
3302                 \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3303             }
3304         }
3305     }
3306 }
3307 }
3308 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3309     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3310 }
3311 % todo notations
3312 }}
3313 }
3314 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3315 \tl_put_left:Nx \l_tmpa_tl {
3316     \prop_set_from_keyval:cn {
3317         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3318     }{
3319         \prop_to_keyval:N \l_stex_current_copymodule_prop
3320     }
3321 }
3322 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3323 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3324 \exp_args:Nx \stex_do_aftergroup:n {
3325     \exp_args:No \exp_not:n \l_tmpa_tl
3326 }
3327 \stex_if_smsmode:F {
3328     \end{stex_annotate_env}
3329 }
3330 }
3331
3332 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3333     \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3334     \stex_deactivate_macro:Nn \symdecl {module~environments}
3335     \stex_deactivate_macro:Nn \symdef {module~environments}
3336     \stex_deactivate_macro:Nn \notation {module~environments}
3337     \stex_reactivate_macro:N \assign
3338     \stex_reactivate_macro:N \renamedec1
3339     \stex_reactivate_macro:N \donotcopy
3340     \stex_smsmode_do:
3341 }{
3342     \stex_copymodule_end:n {}
3343 }
3344
3345 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{

```

```

3346 \stex_copymodule_start:nnnn { #1 } { #2 } { #3 } { realization }
3347 \stex_deactivate_macro:Nn \symdecl {module~environments}
3348 \stex_deactivate_macro:Nn \symdef {module~environments}
3349 \stex_deactivate_macro:Nn \notation {module~environments}
3350 \stex_reactivate_macro:N \assign
3351 \stex_reactivate_macro:N \renamedec1
3352 \stex_reactivate_macro:N \donotcopy
3353 \stex_smsmode_do:
3354 }{
3355 \stex_copymodule_end:n {
3356 \tl_if_exist:cF {
3357 l__stex_features_copymodule_##1?##2_def_tl
3358 }{
3359 \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3360 ##1?##2
3361 }{\l_stex_current_copymodule_name_str}
3362 }
3363 }
3364 }
3365
3366 \NewDocumentCommand \donotcopy { 0{} m }{
3367 \stex_import_module_uri:nn { #1 } { #2 }
3368 \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3369 \seq_map_inline:Nn \l_stex_collect_imports_seq {
3370 \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3371 \seq_map_inline:cn {c_stex_module_##1_constants}{
3372 \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3373 \bool_lazy_any_p:nT {
3374 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?#####1_name_str}}
3375 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?#####1_macroname_str}}
3376 { \cs_if_exist_p:c {l__stex_features_copymodule_##1?#####1_def_tl}}
3377 }{
3378 % TODO throw error
3379 }
3380 }
3381 }
3382
3383 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3384 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3385 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3386 }
3387
3388 \NewDocumentCommand \assign { m m }{
3389 \stex_get_symbol_in_copymodule:n {#1}
3390 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3391 \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
3392 }
3393
3394 \keys_define:nn { stex / renamedec1 } {
3395 name .str_set_x:N = \l_stex_renamedec1_name_str
3396 }
3397 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3398 \str_clear:N \l_stex_renamedec1_name_str
3399

```

```

3400 \keys_set:nn { stex / renamedec1 } { #1 }
3401 }
3402
3403 \NewDocumentCommand \renamedec1 { 0{} m m}{
3404 \__stex_features_renamedec1_args:n { #1 }
3405 \stex_get_symbol_in_copymodule:n {#2}
3406 \stex_debug:nn{renamedec1}{renaming-{\l_stex_get_symbol_uri_str}-to~#3}
3407 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3408 \str_if_empty:NTF \l_stex_renamedec1_name_str {
3409 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3410 \l_stex_get_symbol_uri_str
3411 } }
3412 } {
3413 \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3414 \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3415 \prop_set_eq:cc {l_stex_symdecl_
3416 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3417 _prop
3418 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3419 \seq_set_eq:cc {l_stex_symdecl_
3420 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3421 _notations
3422 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3423 \prop_put:cnx {l_stex_symdecl_
3424 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3425 _prop
3426 }{ name }{ \l_stex_renamedec1_name_str }
3427 \prop_put:cnx {l_stex_symdecl_
3428 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3429 _prop
3430 }{ module }{ \l_stex_current_module_str }
3431 \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3432 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3433 }
3434 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3435 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3436 } }
3437 }
3438 }
3439 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3440 % \_stex_notation_args:n { #1 }
3441 % \tl_clear:N \l_stex_symdecl_definiens_tl
3442 % \stex_get_symbol_in_copymodule:n { #2 }
3443 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3444 % % todo
3445 %}
3446 \stex_deactivate_macro:Nn \assign {copymodules}
3447 \stex_deactivate_macro:Nn \renamedec1 {copymodules}
3448 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3449
3450
3451 \seq_new:N \l_stex_implicit_morphisms_seq
3452 \NewDocumentCommand \implicitmorphism { 0{} m m}{
3453 \stex_import_module_uri:nn { #1 } { #2 }

```

```

3454 \stex_debug:nn{implicits}{
3455   Implicit~morphism:~
3456   \l_stex_module_ns_str ? \l__stex_features_name_str
3457 }
3458 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3459   \l_stex_module_ns_str ? \l__stex_features_name_str
3460 }{
3461   \msg_error:nnn{stex}{error/conflictingmodules}{
3462     \l_stex_module_ns_str ? \l__stex_features_name_str
3463   }
3464 }
3465
3466 % TODO
3467
3468
3469
3470 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3471   \l_stex_module_ns_str ? \l__stex_features_name_str
3472 }
3473 }
3474

```

32.2 The feature environment

structural@feature

```

3475
3476 \NewDocumentEnvironment{structural@feature}{ m m m }{
3477   \stex_if_in_module:F {
3478     \msg_set:nnn{stex}{error/nomodule}{
3479       Structural~Feature~has~to~occur~in~a~module:\\
3480       Feature~#2~of~type~#1\\
3481       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3482     }
3483     \msg_error:nn{stex}{error/nomodule}
3484   }
3485
3486   \str_set:Nx \l_stex_module_name_str {
3487     \prop_item:Nn \l_stex_current_module_prop
3488       { name } / #2 - feature
3489   }
3490
3491   \str_set:Nx \l_stex_module_ns_str {
3492     \prop_item:Nn \l_stex_current_module_prop
3493       { ns }
3494   }
3495
3496
3497   \str_clear:N \l_tmpa_str
3498   \seq_clear:N \l_tmpa_seq
3499   \tl_clear:N \l_tmpa_tl
3500   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3501     origname = #2,
3502     name      = \l_stex_module_name_str ,
3503     ns        = \l_stex_module_ns_str ,

```

```

3504     imports    = \exp_not:o { \l_tmpa_seq } ,
3505     constants  = \exp_not:o { \l_tmpa_seq } ,
3506     content    = \exp_not:o { \l_tmpa_tl } ,
3507     file       = \exp_not:o { \g_stex_currentfile_seq } ,
3508     lang       = \l_stex_module_lang_str ,
3509     sig        = \l_tmpa_str ,
3510     meta       = \l_tmpa_str ,
3511     feature    = #1 ,
3512 }
3513
3514 \stex_if_smsmode:F {
3515   \begin{stex_annotate_env}{ feature:#1 }{}
3516   \stex_annotate_invisible:nnn{header}{}{ #3 }
3517 }
3518 {}{
3519   \str_set:Nx \l_tmpa_str {
3520     c_stex_feature_
3521     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3522     \prop_item:Nn \l_stex_current_module_prop { name }
3523     _prop
3524   }
3525   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3526   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3527   \stex_if_smsmode:TF {
3528     \exp_args:Nx \stex_add_to_sms:n {
3529       \prop_gset_from_keyval:cn {
3530         c_stex_feature_
3531         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3532         \prop_item:Nn \l_stex_current_module_prop { name }
3533         _prop
3534       } {
3535         origname = #2,
3536         name     = \prop_item:cn { \l_tmpa_str } { name } ,
3537         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
3538         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3539         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3540         content  = \prop_item:cn { \l_tmpa_str } { content } ,
3541         file     = \prop_item:cn { \l_tmpa_str } { file } ,
3542         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3543         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3544         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3545         feature  = \prop_item:cn { \l_tmpa_str } { feature }
3546       }
3547     }
3548   } {
3549     \end{stex_annotate_env}
3550   }
3551 }
3552

```

32.3 Features

structure

```

3553
3554 \prop_new:N \l_stex_all_structures_prop
3555
3556 \keys_define:nn { stex / features / structure } {
3557   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3558 }
3559
3560 \cs_new_protected:Nn \__stex_features_structure_args:n {
3561   \str_clear:N \l__stex_features_structure_name_str
3562   \keys_set:nn { stex / features / structure } { #1 }
3563 }
3564
3565 %\stex_new_feature:nnnn { structure } { 0{} m } {
3566 %   \__stex_features_structure_args:n { ##1 }
3567 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3568 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3569 %   }
3570 %} {
3571 %
3572 %}
3573
3574 \NewDocumentEnvironment{mathstructure}{0{} m }{
3575   \__stex_features_structure_args:n { #1 }
3576   \str_if_empty:NT \l__stex_features_structure_name_str {
3577     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3578   }
3579   \exp_args:Nnnx
3580   \begin{structural@feature}{ structure }
3581     { \l__stex_features_structure_name_str }{}
3582     \seq_clear:N \l_tmpa_seq
3583     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3584     \stex_smsmode_do:
3585   }{
3586     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3587     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3588     \str_set:Nx \l_tmpa_str {
3589       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3590       \prop_item:Nn \l_stex_current_module_prop { name }
3591     }
3592     \seq_map_inline:Nn \l_tmpa_seq {
3593       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3594     }
3595     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3596     \exp_args:Nnx
3597     \AddToHookNext { env / mathstructure / after }{
3598       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3599         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3600       }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3601     \STEXexport {
3602       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3603         {\prop_item:Nn \l_stex_current_module_prop { origname }}
3604         {\l_tmpa_str}
3605       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3606         {#2}{\l_tmpa_str}

```



```

3607 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3608 %      \prop_item:Nn \l_stex_current_module_prop { origname },
3609 %      \l_tmpa_str
3610 %    }
3611 %      \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3612 %      #2,\l_tmpa_str
3613 %    }
3614 %      \tl_set:cx { #2 } {
3615 %      \stex_invoke_structure:n { \l_tmpa_str }
3616 %    }
3617 %  }
3618
3619 \end{structural@feature}
3620 % \g_stex_last_feature_prop
3621 }

```

\instantiate

```

3622 \seq_new:N \l__stex_features_structure_field_seq
3623 \str_new:N \l__stex_features_structure_field_str
3624 \str_new:N \l__stex_features_structure_def_tl
3625 \prop_new:N \l__stex_features_structure_prop
3626 \NewDocumentCommand \instantiate { m O{} m }{
3627   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3628   \prop_set_eq:Nc \l__stex_features_structure_prop {
3629     c_stex_feature_\l_tmpa_str _prop
3630   }
3631   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3632   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3633     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3634     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3635       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3636       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3637         {!} \l_tmpa_tl
3638       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3639         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3640         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3641         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3642       }{
3643         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3644         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3645         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3646           \l_tmpa_tl
3647         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3648           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3649           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3650         }{
3651           \tl_clear:N \l_tmpb_tl
3652         }
3653       }
3654     }{
3655       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3656       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3657         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3658         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl

```

```

3659         \tl_clear:N \l_tmpa_tl
3660     }{
3661         % TODO throw error
3662     }
3663 }
3664 % \l_tmpa_str: name
3665 % \l_tmpa_tl: definiens
3666 % \l_tmpb_tl: notation
3667 \tl_if_empty:NT \l__stex_features_structure_field_str {
3668     % TODO throw error
3669 }
3670 \str_clear:N \l_tmpb_str
3671
3672 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3673 \seq_map_inline:Nn \l_tmpa_seq {
3674     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3675     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3676     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3677         \seq_map_break:n {
3678             \str_set:Nn \l_tmpb_str { ####1 }
3679         }
3680     }
3681 }
3682 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3683     \l_tmpb_str
3684
3685 \tl_if_empty:NTF \l_tmpb_tl {
3686     \tl_if_empty:NF \l_tmpa_tl {
3687         \exp_args:Nx \use:n {
3688             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3689         }
3690     }
3691 }{
3692     \tl_if_empty:NTF \l_tmpa_tl {
3693         \exp_args:Nx \use:n {
3694             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3695         }
3696     }
3697 }{
3698     \exp_args:Nx \use:n {
3699         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3700         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3701     }
3702 }
3703 }
3704 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3705 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3706 % #3/\l__stex_features_structure_field_str
3707 % \par
3708 % \expandafter\present\csname
3709 %     l_stex_symdecl_
3710 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
3711 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3712 % #3/\l__stex_features_structure_field_str

```

```

3713 %      _prop
3714 %      \endcsname
3715 }
3716
3717 \tl_clear:N \l__stex_features_structure_def_tl
3718
3719 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3720 \seq_map_inline:Nn \l_tmpa_seq {
3721   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3722   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3723   \exp_args:Nx \use:n {
3724     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3725
3726     }
3727   }
3728
3729   \prop_if_exist:cF {
3730     l_stex_symdecl_
3731     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3732     \prop_item:Nn \l_stex_current_module_prop {name} ?
3733     #3/\l_tmpa_str
3734     _prop
3735   }{
3736     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3737     \l_tmpb_str
3738     \exp_args:Nx \use:n {
3739       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3740     }
3741   }
3742 }
3743
3744 \symdecl*[type={\STEXsymbol{module-type}}{
3745   \_stex_term_math_oms:nnnn {
3746     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3747     \prop_item:Nn \l__stex_features_structure_prop {name}
3748     }{0}{0}
3749   }]{#3}
3750
3751 % TODO: -> sms file
3752
3753 \tl_set:cx{ #3 }{
3754   \stex_invoke_structure:nnn {
3755     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3756     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3757   } {
3758     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3759     \prop_item:Nn \l__stex_features_structure_prop {name}
3760   }
3761 }
3762 \stex_smsmode_do:
3763 }

```

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

\stex_invoke_structure:nnn

```

3764 % #1: URI of the instance
3765 % #2: URI of the instantiated module
3766 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3767   \tl_if_empty:nTF{ #3 }{
3768     \prop_set_eq:Nc \l__stex_features_structure_prop {
3769       c_stex_feature_ #2 _prop
3770     }
3771     \tl_clear:N \l_tmpa_tl
3772     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3773     \seq_map_inline:Nn \l_tmpa_seq {
3774       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3775       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3776       \cs_if_exist:cT {
3777         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3778       }{
3779         \tl_if_empty:NF \l_tmpa_tl {
3780           \tl_put_right:Nn \l_tmpa_tl {,}
3781         }
3782         \tl_put_right:Nx \l_tmpa_tl {
3783           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3784         }
3785       }
3786     }
3787     \exp_args:No \mathstrut \l_tmpa_tl
3788   }{
3789     \stex_invoke_symbol:n{#1/#3}
3790   }
3791 }

```

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

3792 </package>

Chapter 33

STEX -Statements Implementation

```
3793 <*package>
3794
3795 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3796
3797 \protected\def\ignorespacesandpars{
3798   \begingroup\catcode13=10\relax
3799   \@ifnextchar\par{
3800     \endgroup\expandafter\ignorespacesandpars\@gobble
3801   }{
3802     \endgroup
3803   }
3804 }
3805
3806 <@@=stex_statements>
3807
3808   Warnings and error messages
```

`\titleemph`

```
3808 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

`definiendum`

```
3809 \keys_define:nn {stex / definiendum }{
3810   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3811   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3812   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3813 }
3814 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3815   \str_clear:N \l__stex_statements_definiendum_root_str
3816   \tl_clear:N \l__stex_statements_definiendum_post_tl
3817   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3818 \keys_set:nn { stex / definiendum } { #1 }
3819 }
3820 \NewDocumentCommand \definiendum { 0{} m m } {
3821   \__stex_statements_definiendum_args:n { #1 }
3822   \stex_get_symbol:n { #2 }
3823   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3824   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3825     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3826       \tl_set:Nn \l_tmpa_tl { #3 }
3827     } {
3828       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3829       \tl_set:Nn \l_tmpa_tl {
3830         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3831       }
3832     }
3833   } {
3834     \tl_set:Nn \l_tmpa_tl { #3 }
3835   }
3836
3837   % TODO root
3838   \rustex_if:TF {
3839     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3840   } {
3841     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3842   }
3843 }
3844 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3845
3846 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3847
3848 \NewDocumentCommand \definame { 0{} m } {
3849   \__stex_statements_definiendum_args:n { #1 }
3850   % TODO: root
3851   \stex_get_symbol:n { #2 }
3852   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3853   \str_set:Nx \l_tmpa_str {
3854     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3855   }
3856   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3857   \rustex_if:TF {
3858     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3859       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3860     }
3861   } {
3862     \defemph@uri {
3863       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3864     } { \l_stex_get_symbol_uri_str }
3865   }
3866 }
3867 \stex_deactivate_macro:Nn \definame {definition~environments}

```

```

3868
3869 \NewDocumentCommand \Definame { 0{ } m } {
3870   \__stex_statements_definiendum_args:n { #1 }
3871   \stex_get_symbol:n { #2 }
3872   \str_set:Nx \l_tmpa_str {
3873     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3874   }
3875   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3876   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3877   \rustex_if:TF {
3878     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3879       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3880     }
3881   } {
3882     \defemph@uri {
3883       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3884     } { \l_stex_get_symbol_uri_str }
3885   }
3886 }
3887 \stex_deactivate_macro:Nn \Definame {definition-environments}
3888
3889 \NewDocumentCommand \Symname { 0{ } m }{
3890   \stex_symname_args:n { #1 }
3891   \stex_get_symbol:n { #2 }
3892   \str_set:Nx \l_tmpa_str {
3893     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3894   }
3895   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3896   \let\compemph_uri_prev:\compemph@uri
3897   \let\compemph@uri\symrefemph@uri
3898   \exp_args:NNx \use:nn
3899   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3900     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3901     \l_stex_symname_post_str
3902   ] }
3903   \let\compemph@uri\compemph_uri_prev:
3904 }

```

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

sdefinition

```

3905
3906 \keys_define:nn {stex / sdefinition }{
3907   type      .str_set_x:N = \sdefinitiontype,
3908   id        .str_set_x:N = \sdefinitionid,
3909   name      .str_set_x:N = \sdefinitionname,
3910   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3911   title     .tl_set:N     = \sdefinitiontitle
3912 }
3913 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3914   \str_clear:N \sdefinitiontype
3915   \str_clear:N \sdefinitionid
3916   \str_clear:N \sdefinitionname
3917   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

3918 \tl_clear:N \sdefinitiontitle
3919 \keys_set:nn { stex / sdefinition }{ #1 }
3920 }
3921
3922 \NewDocumentEnvironment{sdefinition}{O{}}{
3923   \__stex_statements_sdefinition_args:n{ #1 }
3924   \stex_reactivate_macro:N \definiendum
3925   \stex_reactivate_macro:N \definame
3926   \stex_reactivate_macro:N \Definame
3927   \stex_if_smsmode:F{
3928     \seq_clear:N \l_tmpa_seq
3929     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
3930       \str_if_eq:nnF{ ##1 }{ }{
3931         \stex_get_symbol:n { ##1 }
3932         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3933           \l_stex_get_symbol_uri_str
3934         }
3935       }
3936     }
3937     \exp_args:Nnnx
3938     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
3939     \str_if_empty:NF \sdefinitiontype {
3940       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
3941     }
3942     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
3943     \tl_clear:N \l_tmpa_tl
3944     \clist_map_inline:Nn \l_tmpa_clist {
3945       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3946         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3947       }
3948     }
3949     \tl_if_empty:NTF \l_tmpa_tl {
3950       \__stex_statements_sdefinition_start:
3951     }{
3952       \l_tmpa_tl
3953     }
3954   }
3955   \stex_ref_new_doc_target:n \sdefinitionid
3956   \stex_smsmode_do:
3957 }{
3958   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
3959   \stex_if_smsmode:F {
3960     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
3961     \tl_clear:N \l_tmpa_tl
3962     \clist_map_inline:Nn \l_tmpa_clist {
3963       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3964         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3965       }
3966     }
3967     \tl_if_empty:NTF \l_tmpa_tl {
3968       \__stex_statements_sdefinition_end:
3969     }{
3970       \l_tmpa_tl
3971     }

```



```

3972     \end{stex_annotate_env}
3973   }
3974 }

```

\stexpatchdefinition

```

3975 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3976   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3977     ~(\sdefinitiontitle)
3978   }~}
3979 }
3980 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3981
3982 \newcommand\stexpatchdefinition[3] [] {
3983   \str_set:Nx \l_tmpa_str{ #1 }
3984   \str_if_empty:NTF \l_tmpa_str {
3985     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3986     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3987   }{
3988     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3989     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3990   }
3991 }

```

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

\inlinedef inline:

```

3992 \keys_define:nn {stex / inlinedef }{
3993   type      .str_set_x:N = \sdefinitiontype,
3994   id        .str_set_x:N = \sdefinitionid,
3995   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
3996   name      .str_set_x:N = \sdefinitionname
3997 }
3998 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
3999   \str_clear:N \sdefinitiontype
4000   \str_clear:N \sdefinitionid
4001   \str_clear:N \sdefinitionname
4002   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4003   \keys_set:nn { stex / inlinedef }{ #1 }
4004 }
4005 \NewDocumentCommand \inlinedef { 0{} m } {
4006   \begingroup
4007   \__stex_statements_inlinedef_args:n{ #1 }
4008   \stex_ref_new_doc_target:n \sdefinitionid
4009   \stex_reactivate_macro:N \definiendum
4010   \stex_reactivate_macro:N \definame
4011   \stex_reactivate_macro:N \Definame
4012   \stex_if_smsmode:TF{
4013     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4014   }{
4015     \seq_clear:N \l_tmpa_seq
4016     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4017       \str_if_eq:nnF{ ##1 }{}{
4018         \stex_get_symbol:n { ##1 }
4019         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4020         \l_stex_get_symbol_uri_str
4021     }
4022 }
4023 }
4024 \exp_args:Nnx
4025 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4026     \str_if_empty:NF \sdefinitiontype {
4027         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4028     }
4029     #2
4030     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4031 }
4032 }
4033 \endgroup
4034 \stex_smsmode_do:
4035 }

```

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

33.2 Assertions

sassertion

```

4036
4037 \keys_define:nn {stex / sassertion }{
4038     type      .str_set_x:N = \sassertiontype,
4039     id        .str_set_x:N = \sassertionid,
4040     title     .tl_set:N    = \sassertiontitle ,
4041     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4042     name      .str_set_x:N = \sassertionname
4043 }
4044 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4045     \str_clear:N \sassertiontype
4046     \str_clear:N \sassertionid
4047     \str_clear:N \sassertionname
4048     \clist_clear:N \l__stex_statements_sassertion_for_clist
4049     \tl_clear:N \sassertiontitle
4050     \keys_set:nn { stex / sassertion }{ #1 }
4051 }
4052
4053 %\tl_new:N \g__stex_statements_aftergroup_tl
4054
4055 \NewDocumentEnvironment{sassertion}{0{}}{
4056     \__stex_statements_sassertion_args:n{ #1 }
4057     \stex_if_smsmode:F {
4058         \seq_clear:N \l_tmpa_seq
4059         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4060             \str_if_eq:nnF{ ##1 }{ }{
4061                 \stex_get_symbol:n { ##1 }
4062                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4063                     \l_stex_get_symbol_uri_str
4064                 }
4065             }
4066         }

```

```

4067 \exp_args:Nnnx
4068 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4069 \str_if_empty:NF \sassertiontype {
4070   \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4071 }
4072 \clist_set:No \l_tmpa_clist \sassertiontype
4073 \tl_clear:N \l_tmpa_tl
4074 \clist_map_inline:Nn \l_tmpa_clist {
4075   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4076     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4077   }
4078 }
4079 \tl_if_empty:NTF \l_tmpa_tl {
4080   \__stex_statements_sassertion_start:
4081 }{
4082   \l_tmpa_tl
4083 }
4084 }
4085 \stex_ref_new_doc_target:n \sassertionid
4086 \stex_smsmode_do:
4087 ){
4088   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4089   \stex_if_smsmode:F {
4090     \clist_set:No \l_tmpa_clist \sassertiontype
4091     \tl_clear:N \l_tmpa_tl
4092     \clist_map_inline:Nn \l_tmpa_clist {
4093       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4094         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4095       }
4096     }
4097     \tl_if_empty:NTF \l_tmpa_tl {
4098       \__stex_statements_sassertion_end:
4099     }{
4100       \l_tmpa_tl
4101     }
4102     \end{stex_annotate_env}
4103   }
4104 }

```

\stexpatchassertion

```

4105
4106 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4107   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4108     (\sassertiontitle)
4109   }~}
4110 }
4111 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4112
4113 \newcommand\stexpatchassertion[3] [] {
4114   \str_set:Nx \l_tmpa_str{ #1 }
4115   \str_if_empty:NTF \l_tmpa_str {
4116     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4117     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4118   }{

```

```

4119     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4120     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4121   }
4122 }

```

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

`\inlineass` inline:

```

4123 \keys_define:nn {stex / inlineass }{
4124   type      .str_set_x:N = \sassertiontype,
4125   id        .str_set_x:N = \sassertionid,
4126   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4127   name      .str_set_x:N = \sassertionname
4128 }
4129 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4130   \str_clear:N \sassertiontype
4131   \str_clear:N \sassertionid
4132   \str_clear:N \sassertionname
4133   \clist_clear:N \l__stex_statements_sassertion_for_clist
4134   \keys_set:nn { stex / inlineass }{ #1 }
4135 }
4136 \NewDocumentCommand \inlineass { 0{} m } {
4137   \begingroup
4138   \__stex_statements_inlineass_args:n{ #1 }
4139   \stex_ref_new_doc_target:n \sassertionid
4140   \stex_if_smsmode:TF{
4141     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4142   }{
4143     \seq_clear:N \l_tmpa_seq
4144     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4145       \str_if_eq:nnF{ ##1 }{}{
4146         \stex_get_symbol:n { ##1 }
4147         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4148           \l_stex_get_symbol_uri_str
4149         }
4150       }
4151     }
4152     \exp_args:Nnx
4153     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4154       \str_if_empty:NF \sassertiontype {
4155         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4156       }
4157       #2
4158       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4159     }
4160   }
4161   \endgroup
4162   \stex_smsmode_do:
4163 }

```

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

33.3 Examples

sexample

```

4164
4165 \keys_define:nn {stex / sexample }{
4166   type      .str_set_x:N = \exampletype,
4167   id        .str_set_x:N = \sexampleid,
4168   title     .tl_set:N     = \sexampletile,
4169   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
4170 }
4171 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4172   \str_clear:N \sexampletype
4173   \str_clear:N \sexampleid
4174   \tl_clear:N \sexampletile
4175   \clist_clear:N \l__stex_statements_sexample_for_clist
4176   \keys_set:nn { stex / sexample }{ #1 }
4177 }
4178
4179 \NewDocumentEnvironment{sexample}{0{}}{
4180   \__stex_statements_sexample_args:n{ #1 }
4181   \stex_if_smsmode:F {
4182     \seq_clear:N \l_tmpa_seq
4183     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4184       \str_if_eq:nnF{ ##1 }{}{
4185         \stex_get_symbol:n { ##1 }
4186         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4187           \l_stex_get_symbol_uri_str
4188         }
4189       }
4190     }
4191     \exp_args:Nnnx
4192     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4193     \str_if_empty:NF \sexampletype {
4194       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4195     }
4196     \clist_set:Nn \l_tmpa_clist \sexampletype
4197     \tl_clear:N \l_tmpa_tl
4198     \clist_map_inline:Nn \l_tmpa_clist {
4199       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4200         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4201       }
4202     }
4203     \tl_if_empty:NTF \l_tmpa_tl {
4204       \__stex_statements_sexample_start:
4205     }{
4206       \l_tmpa_tl
4207     }
4208   }
4209   \stex_ref_new_doc_target:n \sexampleid
4210   \stex_smsmode_do:
4211 }{
4212   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4213   \stex_if_smsmode:F {
4214     \clist_set:Nn \l_tmpa_clist \sexampletype

```

```

4215 \tl_clear:N \l_tmpa_tl
4216 \clist_map_inline:Nn \l_tmpa_clist {
4217   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4218     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4219   }
4220 }
4221 \tl_if_empty:NTF \l_tmpa_tl {
4222   \__stex_statements_sexample_end:
4223 }{
4224   \l_tmpa_tl
4225 }
4226 \end{stex_annotate_env}
4227 }
4228 }

```

\stexpatchexample

```

4229
4230 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4231   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplename {
4232     (\sexamplename)
4233   }~}
4234 }
4235 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4236
4237 \newcommand\stexpatchexample[3] [] {
4238   \str_set:Nx \l_tmpa_str{ #1 }
4239   \str_if_empty:NTF \l_tmpa_str {
4240     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4241     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4242   }{
4243     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4244     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4245   }
4246 }

```

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

\inlineex inline:

```

4247 \keys_define:nn {stex / inlineex }{
4248   type      .str_set_x:N = \sexamplename,
4249   id        .str_set_x:N = \sexampleid,
4250   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4251   name      .str_set_x:N = \sexamplename
4252 }
4253 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4254   \str_clear:N \sexamplename
4255   \str_clear:N \sexampleid
4256   \str_clear:N \sexamplename
4257   \clist_clear:N \l__stex_statements_sexample_for_clist
4258   \keys_set:nn { stex / inlineex }{ #1 }
4259 }
4260 \NewDocumentCommand \inlineex { 0{} m } {
4261   \begin{group}
4262     \__stex_statements_inlineex_args:n{ #1 }

```

```

4263 \stex_ref_new_doc_target:n \sexampleid
4264 \stex_if_smsmode:TF{
4265   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4266 }{
4267   \seq_clear:N \l_tmpa_seq
4268   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4269     \str_if_eq:nnF{ ##1 }{ }{
4270       \stex_get_symbol:n { ##1 }
4271       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4272         \l_stex_get_symbol_uri_str
4273       }
4274     }
4275   }
4276   \exp_args:Nnx
4277   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{ }{
4278     \str_if_empty:NF \sexamplename {
4279       \stex_annotate_invisible:nnn{type}{\sexamplename}{ }
4280     }
4281     #2
4282     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4283   }
4284 }
4285 \endgroup
4286 \stex_smsmode_do:
4287 }

```

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

33.4 Logical Paragraphs

sparagraph

```

4288 \keys_define:nn { stex / sparagraph } {
4289   id      .str_set:N = \sparagraphid ,
4290   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4291   type    .str_set:N  = \sparagraphtype ,
4292   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4293   from    .tl_set:N   = \sparagraphfrom ,
4294   to      .tl_set:N   = \sparagraphto ,
4295   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
4296   name    .str_set:N   = \sparagraphname
4297 }
4298
4299 \cs_new_protected:Nn \stex_sparagraph_args:n {
4300   \tl_clear:N \l_stex_sparagraph_title_tl
4301   \tl_clear:N \sparagraphfrom
4302   \tl_clear:N \sparagraphto
4303   \tl_clear:N \l_stex_sparagraph_start_tl
4304   \str_clear:N \sparagraphid
4305   \str_clear:N \sparagraphtype
4306   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4307   \str_clear:N \sparagraphname
4308   \keys_set:nn { stex / sparagraph }{ #1 }
4309 }

```

```

4310 \newif\if@in@omtext\@in@omtextfalse
4311
4312 \NewDocumentEnvironment {sparagraph} { 0{} } {
4313   \stex_sparagraph_args:n { #1 }
4314   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4315     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4316   }{
4317     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4318   }
4319   \@in@omtexttrue
4320   \stex_if_smsmode:F {
4321     \seq_clear:N \l_tmpa_seq
4322     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4323       \str_if_eq:nnF{ ##1 }{}{
4324         \stex_get_symbol:n { ##1 }
4325         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4326           \l_stex_get_symbol_uri_str
4327         }
4328       }
4329     }
4330     \exp_args:Nnnx
4331     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4332     \str_if_empty:NF \sparagraphtype {
4333       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4334     }
4335     \str_if_empty:NF \sparagraphfrom {
4336       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4337     }
4338     \str_if_empty:NF \sparagraphto {
4339       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4340     }
4341     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4342     \tl_clear:N \l_tmpa_tl
4343     \clist_map_inline:Nn \sparagraphtype {
4344       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4345         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4346       }
4347     }
4348     \tl_if_empty:NTF \l_tmpa_tl {
4349       \__stex_statements_sparagraph_start:
4350     }{
4351       \l_tmpa_tl
4352     }
4353   }
4354   \stex_ref_new_doc_target:n \sparagraphid
4355   \stex_smsmode_do:
4356   \ignorespacesandpars
4357 }{
4358   \stex_if_smsmode:F {
4359     \clist_set:Nn \l_tmpa_clist \sparagraphtype
4360     \tl_clear:N \l_tmpa_tl
4361     \clist_map_inline:Nn \l_tmpa_clist {
4362       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4363         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```



```

4364     }
4365   }
4366   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname} }
4367   \tl_if_empty:NTF \l_tmpa_tl {
4368     \__stex_statements_sparagraph_end:
4369   }{
4370     \l_tmpa_tl
4371   }
4372   \end{stex_annotate_env}
4373 }
4374 }

```

\stexpatchparagraph

```

4375
4376 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4377   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4378     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4379       \titleemph{\l_stex_sparagraph_title_tl}:~
4380     }
4381   }{
4382     \titleemph{\l_stex_sparagraph_start_tl}~
4383   }
4384 }
4385 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4386
4387 \newcommand\stexpatchparagraph[3] [] {
4388   \str_set:Nx \l_tmpa_str{ #1 }
4389   \str_if_empty:NTF \l_tmpa_str {
4390     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4391     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4392   }{
4393     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4394     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4395   }
4396 }
4397
4398 \keys_define:nn { stex / inlinepara} {
4399   id      .str_set_x:N = \sparagraphid ,
4400   type    .str_set_x:N = \sparagraphtype ,
4401   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4402   from    .tl_set:N    = \sparagraphfrom ,
4403   to      .tl_set:N    = \sparagraphto ,
4404   name    .str_set:N    = \sparagraphname
4405 }
4406 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4407   \tl_clear:N \sparagraphfrom
4408   \tl_clear:N \sparagraphto
4409   \str_clear:N \sparagraphid
4410   \str_clear:N \sparagraphtype
4411   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4412   \str_clear:N \sparagraphname
4413   \keys_set:nn { stex / inlinepara }{ #1 }
4414 }
4415 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4416 \begingroup
4417 \__stex_statements_inlinepara_args:n{ #1 }
4418 \stex_ref_new_doc_target:n \sparagraphid
4419 \stex_if_smsmode:TF{
4420   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4421 }{
4422   \seq_clear:N \l_tmpa_seq
4423   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4424     \str_if_eq:nnF{ ##1 }{}{
4425       \stex_get_symbol:n { ##1 }
4426       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4427         \l_stex_get_symbol_uri_str
4428       }
4429     }
4430   }
4431   \exp_args:Nnx
4432   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4433     \str_if_empty:NF \sparagraphtype {
4434       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4435     }
4436     \str_if_empty:NF \sparagraphfrom {
4437       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4438     }
4439     \str_if_empty:NF \sparagraphto {
4440       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4441     }
4442     #2
4443     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4444   }
4445 }
4446 \endgroup
4447 \stex_smsmode_do:
4448 }
4449

```

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

symboldoc

```

4450 \NewDocumentEnvironment{symboldoc}{ m }{
4451   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4452   \seq_clear:N \l_tmpb_seq
4453   \seq_map_inline:Nn \l_tmpa_seq {
4454     \str_if_eq:nnF{ ##1 }{}{
4455       \stex_get_symbol:n { ##1 }
4456       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4457         \l_stex_get_symbol_uri_str
4458       }
4459     }
4460   }
4461   \par
4462   \exp_args:Nnnx
4463   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4464 }{
4465   \end{stex_annotate_env}
4466 }

```

4467 </package>

Chapter 34

The Implementation

34.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).¹³

```
4468 <*package>
4469 <@@=stex_sproof>
4470
4471 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4472
```

34.2 Proofs

We first define some keys for the proof environment.

```
4473 \keys_define:nn { stex / spf } {
4474   id          .str_set:N = \l__stex_sproof_spf_id_str,
4475   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4476   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4477   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4478   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4479   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4480   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4481   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4482   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4483   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4484 }
4485 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4486   \str_clear:N \l__stex_sproof_spf_id_str
4487   \tl_clear:N \l__stex_sproof_spf_display_tl
4488   \tl_clear:N \l__stex_sproof_spf_for_tl
4489   \tl_clear:N \l__stex_sproof_spf_from_tl
4490   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4491   \tl_clear:N \l__stex_sproof_spf_type_tl
4492   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4493 \tl_clear:N \l__stex_sproof_spf_continues_tl
4494 \tl_clear:N \l__stex_sproof_spf_functions_tl
4495 \tl_clear:N \l__stex_sproof_spf_method_tl
4496 \keys_set:nn { stex / spf }{ #1 }
4497 }

```

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

```

4498 \def\spf@flow{flow}

```

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

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

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

```

4499 \newcount\count_ten
4500 \newenvironment{pst@with@label}[1]{
4501   \edef\pst@label{#1}
4502   \advance\count_ten by 1\relax
4503   \count_ten=1
4504 }{
4505   \advance\count_ten by -1\relax
4506 }

```

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

```

4507 \def\the@pst@label{
4508   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4509 }

```

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

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

```

4510 \keys_define:nn { stex / pstlabel }{
4511   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4512   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4513   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4514 }
4515 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4516 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4517 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4518 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4519 }
4520 \__stex_sproof_pstlabel_args:n {}
4521 \newcommand\setpstlabelstyle[1]{
4522   \__stex_sproof_pstlabel_args:n {#1}
4523 }
4524 \newcommand\setpstlabelstyledefault{%
4525   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4526 }

```

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

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

```

4527 \ExplSyntaxOff
4528 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4529 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4530 \def\pst@make@label@short#1#2{#2}
4531 \def\pst@make@label@empty#1#2{}
4532 \ExplSyntaxOn
4533 \def\pstlabelstyle#1{%
4534   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4535 }%
4536 \pstlabelstyle{long}%

```

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

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

```

4537 \def\next@pst@label{%
4538   \global\advance\count\count10 by 1%
4539 }%

```

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

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

```

4540 \def\sproof@box{
4541   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4542 }
4543 \def\spf@proofend{\sproof@box}
4544 \def\sproofend{
4545   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4546     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4547   }
4548 }
4549 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4550 \def\spf@proofsketch@kw{Proof Sketch}
4551 \def\spf@proof@kw{Proof}
4552 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4553 \AddToHook{begindocument}{
4554   \ltx@ifpackageloaded{babel}{
4555     \makeatletter
4556     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4557     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4558       \input{sproof-ngerman.ldf}
4559     }
4560     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4561       \input{sproof-finnish.ldf}
4562     }
4563     \clist_if_in:NnT \l_tmpa_clist {french}{
4564       \input{sproof-french.ldf}
4565     }
4566     \clist_if_in:NnT \l_tmpa_clist {russian}{
4567       \input{sproof-russian.ldf}
4568     }
4569     \makeatother
4570   }{}
4571 }

```

`spfsketch`

```

4572 \newcommand\spfsketch[2][]{
4573   \__stex_sproof_spf_args:n{#1}
4574   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4575     \titleemph{
4576       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4577         \spf@proofsketch@kw
4578       }{
4579         \l__stex_sproof_spf_type_tl
4580       }
4581     }:
4582   }
4583   {~#2}
4584   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4585   \sproofend
4586 }

```

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

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

```

4587 \newenvironment{spfeq}[2][]{
4588   \__stex_sproof_spf_args:n{#1}
4589   %\sref@target
4590   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4591     \titleemph{
4592       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4593         \spf@proof@kw
4594       }{

```

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

¹⁵EDNOTE: document above

```

4595         \l__stex_sproof_spf_type_tl
4596     }
4597     }:
4598 }
4599 {-#2}
4600 \begin{displaymath}\begin{array}{rcll}
4601 }{
4602 \end{array}\end{displaymath}
4603 }

```

(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.

```

4604 \newenvironment{spf@proof}[2] []{
4605   \l__stex_sproof_spf_args:n{#1}
4606   %\sref@target
4607   \count_ten=10
4608   \par\noindent
4609   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4610     \titleemph{
4611       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4612         \spf@proof@kw
4613       }{
4614         \l__stex_sproof_spf_type_tl
4615       }
4616     }:
4617   }
4618   {-#2}
4619   %\sref@label{id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4620   \def\pst@label{}
4621   \newcount\pst@count% initialize the labeling mechanism
4622   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4623   }{
4624     \end{pst@with@label}\end{description}
4625   }
4626   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4627   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4628 \newcommand\spfidea[2] []{
4629   \l__stex_sproof_spf_args:n{#1}
4630   \titleemph{
4631     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4632       \l__stex_sproof_spf_type_tl
4633     }:
4634   }-#2
4635   \sproofend
4636 }

```

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

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

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

EdN:16

```

spfstep 16
4637 \newenvironment{spfstep}[1][]{
4638   \_stex_sproof_spf_args:n{#1}
4639   \@in@omtexttrue
4640   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4641     \item[\the@pst@label]
4642   }
4643   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4644     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4645   }
4646   %\sref@label{id{\pst@label}
4647   \ignorespacesandpars
4648 }{
4649   \next@pst@label\ignorespacesandpars
4650 }

```

sproofcomment

```

4651 \newenvironment{sproofcomment}[1][]{
4652   \_stex_sproof_spf_args:n{#1}
4653   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4654     \item[\the@pst@label]
4655   }
4656 }{
4657   \next@pst@label
4658 }

```

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

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

```

4659 \newenvironment{subproof}[2][]{
4660   \_stex_sproof_spf_args:n{#1}
4661   \def\@test{#2}
4662   \ifx\@test\empty\else
4663     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4664       \item[\the@pst@label]
4665     }{#2}
4666   \fi
4667   \begin{pst@with@label}{\pst@label,\number\count_ten}
4668 }{
4669   \end{pst@with@label}\next@pst@label
4670 }

```

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

```

4671 \newenvironment{spfcases}[2][]{
4672   \def\@test{#1}
4673   \ifx\@test\empty
4674     \begin{subproof}[method=by-cases]{#2}

```

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

```

4675 \else
4676   \begin{subproof}[#1,method=by-cases]{#2}
4677 \fi
4678 }{
4679   \end{subproof}
4680 }

```

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

```

4681 \newenvironment{spfcase}[2] [] {
4682   \__stex_sproof_spf_args:n{#1}
4683   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4684     \item[\the@pst@label]
4685   }
4686   \def\@test{#2}
4687   \ifx\@test\@empty
4688   \else
4689     {\titleemph{#2}:~}
4690   \fi
4691   \begin{pst@with@label}{\pst@label,\number\count_ten}
4692 }{
4693   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4694     \sproofend
4695   }
4696   \end{pst@with@label}
4697   \next@pst@label
4698 }

```

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

```

4699 \newcommand\spfcasesketch[3] [] {
4700   \__stex_sproof_spf_args:n{#1}
4701   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4702     \item[\the@pst@label]
4703   }
4704   \def\@test{#2}
4705   \ifx\@test\@empty
4706   \else
4707     {\titleemph{#2}:~}
4708   \fi#3
4709   \next@pst@label
4710 }%

```

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.

```

4711 \keys_define:nn { stex / just }{
4712   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4713   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4714   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4715   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4716 }

```

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

`justification`

4717 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4718 `\newcommand\premise[2] [] {#2}`

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

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

4719 `\newcommand\justarg[2] [] {#2}`

4720 `\</package>`

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

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

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

Chapter 35

STEX -Others Implementation

```
4721 <*package>
4722
4723 %%%%%%%%%% others.dtx %%%%%%%%%%
4724
4725 <@@=stex_others>
    Warnings and error messages
4726 % None

\MSC Math subject classifier

4727 \NewDocumentCommand \MSC {m} {
4728 % TODO
4729 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4730 \@ifpackageloaded{tikzinput}{
4731 \RequirePackage{stex-tikzinput}
4732 }{}
4733 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4734 <*package>
4735 <@@=stex_modules>
4736
4737 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4738
4739 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4740 \begingroup
4741 \stex_module_setup:nn{
4742   ns=\c_stex_metatheory_ns_str,
4743   meta=NONE
4744 }{Metatheory}
4745 \stex_reactivate_macro:N \symdecl
4746 \stex_reactivate_macro:N \notation
4747 \stex_reactivate_macro:N \symdef
4748 \ExplSyntaxOff
4749 \csname stex_suppress_html:n\endcsname{
4750   % is-a (a:A, a \in A, a is an A, etc.)
4751   \symdecl[args=ai]{isa}
4752   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4753   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4754   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4755
4756   % bind (\forall, \Pi, \lambda etc.)
4757   \symdecl[args=Bi]{bind}
4758   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4759   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4760   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
4761
4762   % dummy variable
4763   \symdecl{dummyvar}
4764   \notation[underscore]{dummyvar}{\comp\_}
4765   \notation[dot]{dummyvar}{\comp\cdot}
4766   \notation[dash]{dummyvar}{\comp{\rm --}}
4767
4768   %fromto (function space, Hom-set, implication etc.)
```

```

4769 \symdecl[args=ai]{fromto}
4770 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4771 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4772
4773 % mapto (lambda etc.)
4774 %\symdecl[args=Bi]{mapto}
4775 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4776 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4777 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4778
4779 % function/operator application
4780 \symdecl[args=ia]{apply}
4781 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4782 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4783
4784 % ‘‘type’’ of all collections (sets, classes, types, kinds)
4785 \symdecl{collection}
4786 \notation[U]{collection}{\comp{\mathcal{U}}}
4787 \notation[set]{collection}{\comp{\textsf{Set}}}
4788
4789 % sequences
4790 \symdecl[args=1]{seqtype}
4791 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4792
4793 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4794 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4795
4796 %\symdef[args=3,li]{sequence-from-to}{#1}_{#2}\comp{\,\ellipses\,}#1_{#3}
4797 %\notation[ui]{sequence-from-to}{#1}^{#2}\comp{\,\ellipses\,}#1^{#3}
4798 % ^ superceded by \aseqfromto and \livar/\uivar
4799
4800 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
4801 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
4802 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}}{#1\comp,#2}
4803
4804 % letin (‘‘let’’, local definitions, variable substitution)
4805 \symdecl[args=bii]{letin}
4806 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
4807 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4808 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4809
4810 % structures
4811 \symdecl*[args=1]{module-type}
4812 \notation{module-type}{\mathtt{MOD} #1}
4813 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4814 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4815
4816 }
4817 \ExplSyntaxOn
4818 \stex_add_to_current_module:n{
4819   \let\nappa\apply
4820   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4821   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4822   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4823 \def\uivar{\csname sequence-index\endcsname[ui]}
4824 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4825 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4826 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4827 }
4828 \__stex_modules_end_module:
4829 \endgroup
4830 \</package>

```

Chapter 37

Tikzinput Implementation

```
4831 <*package>
4832
4833 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4834
4835 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4836 \RequirePackage{l3keys2e}
4837
4838 \keys_define:nn { tikzinput } {
4839   image .bool_set:N = \c_tikzinput_image_bool,
4840   image .default:n = false ,
4841   unknown .code:n = {}
4842 }
4843
4844 \ProcessKeysOptions { tikzinput }
4845
4846 \bool_if:NTF \c_tikzinput_image_bool {
4847   \RequirePackage{graphicx}
4848
4849   \providecommand\usetikzlibrary[]{}
4850   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4851 }{
4852   \RequirePackage{tikz}
4853   \RequirePackage{standalone}
4854
4855   \newcommand \tikzinput [2] [] {
4856     \setkeys{Gin}{#1}
4857     \ifx \Gin@ewidth \Gin@exclamation
4858       \ifx \Gin@eheight \Gin@exclamation
4859         \input { #2 }
4860       \else
4861         \resizebox{!}{ \Gin@eheight }{
4862           \input { #2 }
4863         }
4864       \fi
4865     \else
4866       \ifx \Gin@eheight \Gin@exclamation
4867         \resizebox{ \Gin@ewidth }{!}{
4868           \input { #2 }
```



```

4869     }
4870     \else
4871         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4872             \input { #2 }
4873         }
4874     \fi
4875 \fi
4876 }
4877 }
4878
4879 \newcommand \ctikzinput [2] [] {
4880     \begin{center}
4881         \tikzinput [1] {#2}
4882     \end{center}
4883 }
4884
4885 \@ifpackageloaded{stex}{
4886     \RequirePackage{stex-tikzinput}
4887 }{}
4888
4889 \</package>
4890 \<*stex>
4891 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4892 \RequirePackage{stex}
4893 \RequirePackage{tikzinput}
4894
4895 \newcommand\mhtikzinput [2] [] {%
4896     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4897     \stex_in_repository:nn\Gin@mhrepos{
4898         \tikzinput [1]{\mhpath{##1}{#2}}
4899     }
4900 }
4901 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4902 \</stex>

```

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

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

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

```
4903 \*cls)
4904 \@@=document_structure)
4905 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4906 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

38.2 Class Options

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

`\omdoc@cls@class`

```
4907 \keys_define:nn{ document-structure / pkg }{
4908   class      .str_set_x:N = \c_document_structure_class_str,
4909   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4910   report     .code:n      = {
4911     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
4912     \str_set:Nn \c_document_structure_class_str {report}
4913   },
4914   book       .code:n      = {
4915     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
4916     \str_set:Nn \c_document_structure_class_str {book}
4917   },
4918   bookpart   .code:n      = {
4919     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
4920     \str_set:Nn \c_document_structure_class_str {book}
4921     \str_set:Nn \c_document_structure_topsect_str {chapter}
4922   },
```

```

4923 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4924 unknown     .code:n      = {
4925   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
4926 }
4927 }
4928 \ProcessKeysOptions{ document-structure / pkg }
4929 \str_if_empty:NT \c_document_structure_class_str {
4930   \str_set:Nn \c_document_structure_class_str {article}
4931 }
4932 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4933   {\c_document_structure_class_str}
4934

```

38.3 Beefing up the document environment

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

```

4935 \RequirePackage{document-structure}
4936 \bool_if:NF \c_document_structure_minimal_bool {

```

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

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

```

4937 \keys_define:nn { document-structure / document }{
4938   id .str_set_x:N = \c_document_structure_document_id_str
4939 }
4940 \let\__document_structure_orig_document=\document
4941 \renewcommand{\document}[1][]{
4942   \keys_set:nn{ document-structure / document }{ #1 }
4943   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4944   \__document_structure_orig_document
4945 }

```

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

```

4946 }
4947 \</cls>

```

38.4 Implementation: document-structure Package

```

4948 \<*package>
4949 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
4950 \RequirePackage{expl-keystr-compat,13keys2e}

```

38.5 Package Options

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

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

```

4951
4952 \keys_define:nn{ document-structure / pkg }{
4953   class      .str_set_x:N = \c_document_structure_class_str,
4954   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4955   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4956 }
4957 \ProcessKeysOptions{ document-structure / pkg }
4958 \str_if_empty:NT \c_document_structure_class_str {
4959   \str_set:Nn \c_document_structure_class_str {article}
4960 }
4961 \str_if_empty:NT \c_document_structure_topsect_str {
4962   \str_set:Nn \c_document_structure_topsect_str {section}
4963 }

```

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

```

4964 \RequirePackage{xspace}
4965 \RequirePackage{comment}
4966 \AddToHook{begindocument}{
4967   \ltx@ifpackageloaded{babel}{
4968     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4969     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4970       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
4971     }
4972   }{}
4973 }

```

`\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.

```

4974 \int_new:N \l_document_structure_section_level_int
4975 \str_case:VnF \c_document_structure_topsect_str {
4976   {part}}{
4977     \int_set:Nn \l_document_structure_section_level_int {0}
4978   }
4979   {chapter}}{
4980     \int_set:Nn \l_document_structure_section_level_int {1}
4981   }
4982 }{
4983   \str_case:VnF \c_document_structure_class_str {
4984     {book}}{
4985       \int_set:Nn \l_document_structure_section_level_int {0}
4986     }
4987     {report}}{
4988       \int_set:Nn \l_document_structure_section_level_int {0}
4989     }
4990   }{
4991     \int_set:Nn \l_document_structure_section_level_int {2}
4992   }
4993 }

```

38.6 Document Structure

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

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

EdN:19

```
4994 \def\current@section@level{document}%
4995 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4996 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4997 \cs_new_protected:Npn \skipomgroup {
4998   \ifcase\l_document_structure_section_level_int
4999   \or\stepcounter{part}
5000   \or\stepcounter{chapter}
5001   \or\stepcounter{section}
5002   \or\stepcounter{subsection}
5003   \or\stepcounter{subsubsection}
5004   \or\stepcounter{paragraph}
5005   \or\stepcounter{subparagraph}
5006   \fi
5007 }
```

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

`blindomgroup`

```
5008 \newcommand\at@begin@blindomgroup[1]{%
5009 \newenvironment{blindomgroup}
5010 {
5011   \int_incr:N\l_document_structure_section_level_int
5012   \at@begin@blindomgroup\l_document_structure_section_level_int
5013 }{}}
```

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

```
5014 \newcommand\omgroup@nonum[2]{
5015   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5016   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5017 }
```

(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.

```
5018 \newcommand\omgroup@num[2]{
```

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

```

5019 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5020   \@nameuse{#1}{#2}
5021 }{
5022   \cs_if_exist:NTF\rdfmata@sectioning{
5023     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5024   }{
5025     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5026   }
5027 }
5028 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5029 }

```

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

omgroup

```

5030 \keys_define:nn { document-structure / omgroup }{
5031   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5032   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5033   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
5034   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5035   srccite      .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5036   type         .tl_set:N    = \l__document_structure_omgroup_type_tl,
5037   short        .tl_set:N    = \l__document_structure_omgroup_short_tl,
5038   display      .tl_set:N    = \l__document_structure_omgroup_display_tl,
5039   intro        .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5040   loadmodules  .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5041 }
5042 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5043   \str_clear:N \l__document_structure_omgroup_id_str
5044   \str_clear:N \l__document_structure_omgroup_date_str
5045   \clist_clear:N \l__document_structure_omgroup_creators_clist
5046   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5047   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5048   \tl_clear:N \l__document_structure_omgroup_type_tl
5049   \tl_clear:N \l__document_structure_omgroup_short_tl
5050   \tl_clear:N \l__document_structure_omgroup_display_tl
5051   \tl_clear:N \l__document_structure_omgroup_intro_tl
5052   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5053   \keys_set:nn { document-structure / omgroup } { #1 }
5054 }

```

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.

```

5055 \newif\if@mainmatter\@mainmattertrue
5056 \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.

```

5057 \keys_define:nn { document-structure / sectioning }{
5058   name .str_set_x:N = \l__document_structure_sect_name_str ,
5059   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5060   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5061   num .bool_set:N = \l__document_structure_sect_num_bool ,
5062 }

```

```

5063 \cs_new_protected:Nn \l__document_structure_sect_args:n {
5064   \str_clear:N \l__document_structure_sect_name_str
5065   \str_clear:N \l__document_structure_sect_ref_str
5066   \bool_set_false:N \l__document_structure_sect_clear_bool
5067   \bool_set_false:N \l__document_structure_sect_num_bool
5068   \keys_set:nn { document-structure / sectioning } { #1 }
5069 }
5070 \newcommand\omdoc@sectioning[3][]{
5071   \l__document_structure_sect_args:n {#1}
5072   \let\omdoc@sect@name\l__document_structure_sect_name_str
5073   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5074   \if@mainmatter% numbering not overridden by frontmatter, etc.
5075     \bool_if:NTF \l__document_structure_sect_num_bool {
5076       \omgroup@num{#2}{#3}
5077     }{
5078       \omgroup@nonum{#2}{#3}
5079     }
5080     \def\current@section@level{\omdoc@sect@name}
5081   \else
5082     \omgroup@nonum{#2}{#3}
5083   \fi
5084 }% 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.

```

5085 \newcommand\omgroup@redefine@addtocontents[1]{%
5086   %\edef\__document_structureimport{#1}%
5087   %\@for\@I:=\__document_structureimport\do{%
5088     %\edef\@path{\csname module@\@I @path\endcsname}%
5089     %\@ifundefined{tf@toc}\relax%
5090     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5091   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5092   %\def\addcontentsline##1##2##3{%
5093     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5094   %\else% hyperref.sty not loaded
5095   %\def\addcontentsline##1##2##3{%
5096     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5097   %\fi
5098 }% hyperref.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.

```

5099 \int_new:N \l__document_structure_omgroup_level_int
5100 \newenvironment{omgroup}[2][]{% keys, title
5101 {
5102   \l__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.

```

5103 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5104   \omgroup@redefine@addtocontents{
5105     %\@ifundefined{module@id}\used@modules%
5106     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5107     }
5108 }

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

5109 \int_incr:N \l_document_structure_omgroup_level_int
5110 \int_incr:N \l_document_structure_section_level_int
5111 \ifcase\l_document_structure_section_level_int
5112   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5113   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5114   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5115   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5116   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5117   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5118   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5119 \fi
5120 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5121 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5122 }% for customization
5123 {}

```

and finally, we localize the sections

```

5124 \newcommand\omdoc@part@kw{Part}
5125 \newcommand\omdoc@chapter@kw{Chapter}
5126 \newcommand\omdoc@section@kw{Section}
5127 \newcommand\omdoc@subsection@kw{Subsection}
5128 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5129 \newcommand\omdoc@paragraph@kw{paragraph}
5130 \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`

```

5131 \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).

```

5132 \cs_if_exist:NTF\frontmatter{
5133   \let\__document_structure_orig_frontmatter\frontmatter
5134   \let\frontmatter\relax
5135 }{
5136   \tl_set:Nn\__document_structure_orig_frontmatter{
5137     \clearpage
5138     \@mainmatterfalse
5139     \pagenumbering{roman}
5140   }
5141 }

```



```

5142 \cs_if_exist:NTF\backmatter{
5143   \let\__document_structure_orig_backmatter\backmatter
5144   \let\backmatter\relax
5145 }{
5146   \tl_set:Nn\__document_structure_orig_backmatter{
5147     \clearpage
5148     \@mainmatterfalse
5149     \pagenumbering{roman}
5150   }
5151 }

```

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.

```

5152 \newenvironment{frontmatter}{
5153   \__document_structure_orig_frontmatter
5154 }{
5155   \cs_if_exist:NTF\mainmatter{
5156     \mainmatter
5157   }{
5158     \clearpage
5159     \@mainmattertrue
5160     \pagenumbering{arabic}
5161   }
5162 }

```

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

```

5163 \newenvironment{backmatter}{
5164   \__document_structure_orig_backmatter
5165 }{
5166   \cs_if_exist:NTF\mainmatter{
5167     \mainmatter
5168   }{
5169     \clearpage
5170     \@mainmattertrue
5171     \pagenumbering{arabic}
5172   }
5173 }

```

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

```

5174 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5175 \def \c__document_structure_document_str{document}
5176 \newcommand\afterprematurestop{}
5177 \def\prematurestop@endomgroup{
5178   \unless\ifx\@currenvir\c__document_structure_document_str
5179     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
5180     \expandafter\prematurestop@endomgroup
5181   \fi
5182 }

```

```

5183 \providecommand\prematurestop{
5184   \message{Stopping~sTeX~processing~prematurely}
5185   \prematurestop@endomgroup
5186   \afterprematurestop
5187   \end{document}
5188 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5189 \RequirePackage{etoolbox}
5190 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5191 \newrobustcmd\useSGvar[1]{%
5192   \@ifundefined{sTeX@Gvar@#1}
5193   {\PackageError{document-structure}
5194     {The sTeX Global variable #1 is undefined}
5195     {set it with \protect\setSGvar}}
5196   \@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.

```

5197 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5198   \@ifundefined{sTeX@Gvar@#1}
5199   {\PackageError{document-structure}
5200     {The sTeX Global variable #1 is undefined}
5201     {set it with \protect\setSGvar}}
5202   {\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.

```
5203 \*cls)
5204 \@@=notesslides)
5205 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5206 \RequirePackage{l3keys2e,expl-keystr-compatible}
5207
5208 \keys_define:nn{notesslides / cls}{
5209   class .code:n = {
5210     \PassOptionsToClass{\CurrentOption}{omdoc}
5211     \str_if_eq:nnT{#1}{book}{
5212       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5213     }
5214     \str_if_eq:nnT{#1}{report}{
5215       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5216     }
5217   },
5218   notes .bool_set:N = \c__notesslides_notes_bool ,
5219   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5220   unknown .code:n = {
5221     \PassOptionsToClass{\CurrentOption}{omdoc}
5222     \PassOptionsToClass{\CurrentOption}{beamer}
5223     \PassOptionsToPackage{\CurrentOption}{notesslides}
5224   }
5225 }
5226 \ProcessKeysOptions{ notesslides / cls }
5227 \bool_if:NTF \c__notesslides_notes_bool {
5228   \PassOptionsToPackage{notes=true}{notesslides}
5229 }{
5230   \PassOptionsToPackage{notes=false}{notesslides}
5231 }
5232 \</cls)
```

now we do the same for the notesslides package.

```

5233 \*package>
5234 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5235 \RequirePackage{13keys2e,expl-keystr-compat}
5236
5237 \keys_define:nn{notesslides / pkg}{
5238   topsect      .str_set_x:N = \c_notesslides_topsect_str,
5239   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
5240   notes        .bool_set:N = \c_notesslides_notes_bool ,
5241   slides        .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
5242   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
5243   frameimages .bool_set:N = \c_notesslides_frameimages_bool ,
5244   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
5245   nopproblems .bool_set:N = \c_notesslides_nopproblems_bool,
5246   unknown      .code:n      = {
5247     \PassOptionsToClass{\CurrentOption}{stex}
5248     \PassOptionsToClass{\CurrentOption}{tikzinput}
5249   }
5250 }
5251 \ProcessKeysOptions{ notesslides / pkg }
5252 \newif\ifnotes
5253 \bool_if:NTF \c_notesslides_notes_bool {
5254   \notesttrue
5255 }{
5256   \notesfalse
5257 }
5258

```

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

```

5259 \str_if_empty:NTF \c_notesslides_topsect_str {
5260   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
5261 }{
5262   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
5263 }
5264 \</package>

```

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

```

5265 \*cls>
5266 \bool_if:NTF \c_notesslides_notes_bool {
5267   \LoadClass{document-structure}
5268 }{
5269   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5270   \newcounter{Item}
5271   \newcounter{paragraph}
5272   \newcounter{subparagraph}
5273   \newcounter{Hfootnote}
5274   \RequirePackage{document-structure}
5275 }

```

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

```

5276 \RequirePackage{notesslides}
5277 \</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.

```

5278 \*package>
5279 \bool_if:NT \c__notesslides_notes_bool {
5280   \RequirePackage{a4wide}
5281   \RequirePackage{marginnote}
5282   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5283   \RequirePackage{mdframed}
5284   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5285   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5286 }
5287 \RequirePackage{stex-tikzinput}
5288 \RequirePackage{etoolbox}
5289 \RequirePackage{amssymb}
5290 \RequirePackage{amsmath}
5291 \RequirePackage{comment}
5292 \RequirePackage{textcomp}
5293 \RequirePackage{url}
5294 \RequirePackage{graphicx}
5295 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.²⁰

```

5296 \bool_if:NT \c__notesslides_notes_bool {
5297   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5298 }

```

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

```

5299 \newcounter{slide}
5300 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5301 \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.

```

5302 \bool_if:NTF \c__notesslides_notes_bool {
5303   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
5304 }{
5305   \excludecomment{note}
5306 }

```

²⁰EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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

```
5307 \bool_if:NT \c__notesslides_notes_bool {
5308   \newlength{\slideframewidth}
5309   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5310 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5311   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5312     \bool_set_true:N #1
5313   }{
5314     \bool_set_false:N #1
5315   }
5316 }
5317 \keys_define:nn{notesslides / frame}{
5318   label .str_set_x:N = \l__notesslides_frame_label_str,
5319   allowframebreaks .code:n = {
5320     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5321   },
5322   allowdisplaybreaks .code:n = {
5323     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5324   },
5325   fragile .code:n = {
5326     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5327   },
5328   shrink .code:n = {
5329     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5330   },
5331   squeeze .code:n = {
5332     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5333   },
5334   t .code:n = {
5335     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5336   },
5337 }
5338 \cs_new_protected:Nn \__notesslides_frame_args:n {
5339   \str_clear:N \l__notesslides_frame_label_str
5340   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5341   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5342   \bool_set_true:N \l__notesslides_frame_fragile_bool
5343   \bool_set_true:N \l__notesslides_frame_shrink_bool
5344   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5345   \bool_set_true:N \l__notesslides_frame_t_bool
5346   \keys_set:nn { notesslides / frame }{ #1 }
5347 }
```

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

```
5348 \renewenvironment{frame}[1][]{
5349   \__notesslides_frame_args:n{#1}
5350   \sffamily
5351   \stepcounter{slide}
5352   \def\@currentlabel{\theslide}
5353   \str_if_empty:NF \l__notesslides_frame_label_str {
5354     \label{\l__notesslides_frame_label_str}
```

5355 }

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

```

5356 \def\itemize@level{outer}
5357 \def\itemize@outer{outer}
5358 \def\itemize@inner{inner}
5359 \renewcommand\newpage{\addtocounter{framenum}{1}}
5360 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5361 \renewenvironment{itemize}{
5362   \ifx\itemize@level\itemize@outer
5363     \def\itemize@label{$\rhd$}
5364   \fi
5365   \ifx\itemize@level\itemize@inner
5366     \def\itemize@label{$\scriptstyle\rhd$}
5367   \fi
5368   \begin{list}
5369   {\itemize@label}
5370   {\setlength{\labelsep}{.3em}
5371    \setlength{\labelwidth}{.5em}
5372    \setlength{\leftmargin}{1.5em}
5373   }
5374   \edef\itemize@level{\itemize@inner}
5375 }{
5376   \end{list}
5377 }

```

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

```

5378 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5379 }{
5380   \medskip\miko@slidelabel\end{mdframed}
5381 }

```

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

`\frametitle`

```

5382 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5383 }

```

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

EdN:21

`\pause` 21

```

5384 \bool_if:NT \c__notesslides_notes_bool {
5385   \newcommand\pause{}
5386 }

```

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

`nparagraph`

```

5387 \bool_if:NTF \c__notesslides_notes_bool {
5388   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5389 }{
5390   \excludecomment{nparagraph}
5391 }

```

²¹EdNOTE: MK: fake it in notes mode for now

nomgroup

```
5392 \bool_if:NTF \c__notesslides_notes_bool {  
5393   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
5394 }{  
5395   \excludecomment{nomgroup}  
5396 }
```

ndefinition

```
5397 \bool_if:NTF \c__notesslides_notes_bool {  
5398   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}  
5399 }{  
5400   \excludecomment{ndefinition}  
5401 }
```

nassertion

```
5402 \bool_if:NTF \c__notesslides_notes_bool {  
5403   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}  
5404 }{  
5405   \excludecomment{nassertion}  
5406 }
```

nsproof

```
5407 \bool_if:NTF \c__notesslides_notes_bool {  
5408   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
5409 }{  
5410   \excludecomment{nproof}  
5411 }
```

nexample

```
5412 \bool_if:NTF \c__notesslides_notes_bool {  
5413   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}  
5414 }{  
5415   \excludecomment{nexample}  
5416 }
```

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

```
5417 \def\inputref@preskip{\smallskip}  
5418 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
5419 \let\orig@inputref\inputref  
5420 \def\inputref{\@ifstar\ninputref\orig@inputref}  
5421 \newcommand\ninputref[2] [] {  
5422   \bool_if:NT \c__notesslides_notes_bool {  
5423     \orig@inputref[#1]{#2}  
5424   }  
5425 }
```

(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>}`.

```

5426 \newlength{\slidelogoheight}
5427
5428 \bool_if:NTF \c__notesslides_notes_bool {
5429   \setlength{\slidelogoheight}{.4cm}
5430 }{
5431   \setlength{\slidelogoheight}{1cm}
5432 }
5433 \newsavebox{\slidelogo}
5434 \sbox{\slidelogo}{\TeX}
5435 \newrobustcmd{\setslidelogo}[1]{
5436   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5437 }
```

(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.

```

5438 \def\source{Michael Kohlhase}% customize locally
5439 \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.

```

5440 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5441 \newsavebox{\cclogo}
5442 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5443 \newif\ifcchref\cchreffalse
5444 \AtBeginDocument{
5445   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5446 }
5447 \def\licensing{
5448   \ifcchref
5449     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5450   \else
5451     {\usebox{\cclogo}}
5452   \fi
5453 }
5454 \newrobustcmd{\setlicensing}[2][]{
5455   \def@url{#1}
5456   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5457   \ifx@url@empty
5458     \def\licensing{{\usebox{\cclogo}}}
5459   \else
5460     \def\licensing{
```

```

5461     \ifcchref
5462     \href{#1}{\usebox{\cclogo}}
5463     \else
5464     {\usebox{\cclogo}}
5465     \fi
5466   }
5467 \fi
5468 }

```

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

EdN:22

`\slidelabel` Now, we set up the slide label for the article mode.²²

```

5469 \newrobustcmd\miko@slidelabel{
5470   \vbox to \slidelogoheight{
5471     \vss\hbox to \slidewidth
5472     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5473   }
5474 }

```

(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.

```

5475 \def\Gin@mhrepos{}
5476 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5477 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5478 \newrobustcmd\frameimage[2][]{
5479   \stepcounter{slide}
5480   \bool_if:NT \c__notesslides_frameimages_bool {
5481     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5482     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5483     \begin{center}
5484       \bool_if:NTF \c__notesslides_fiboxed_bool {
5485         \fbox{
5486           \ifx\Gin@ewidth\@empty
5487             \ifx\Gin@mhrepos\@empty
5488               \mhgraphics[width=\slidewidth,#1]{#2}
5489             \else
5490               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5491             \fi
5492           \else% Gin@ewidth empty
5493             \ifx\Gin@mhrepos\@empty
5494               \mhgraphics[#1]{#2}
5495             \else
5496               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5497             \fi
5498           \fi% Gin@ewidth empty
5499         }
5500       }{
5501         \ifx\Gin@ewidth\@empty

```

²²EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

5502         \ifx\Gin@mhrepos\empty
5503             \mhgraphics[width=\slidewidth,#1]{#2}
5504         \else
5505             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5506         \fi
5507         \ifx\Gin@mhrepos\empty
5508             \mhgraphics[#1]{#2}
5509         \else
5510             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5511         \fi
5512     \fi% Gin@ewidth empty
5513 }
5514 \end{center}
5515 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5516 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5517 }
5518 } % 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.

```

5519 \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

```

5520 \AddToHook{begindocument}{
5521     \definecolor{green}{rgb}{0,.5,0}
5522     \definecolor{purple}{cmyk}{.3,1,0,.17}
5523 }

```

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`.

```

5524 % \def\STpresent#1{\textcolor{blue}{#1}}
5525 \def\defemph#1{\textcolor{magenta}{#1}}
5526 \def\symrefemph#1{\textcolor{cyan}{#1}}
5527 \def\compemph#1{\textcolor{blue}{#1}}
5528 \def\titleemph#1{\textcolor{blue}{#1}}
5529 \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.

```

5530 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5531 \def\smalltextwarning{
5532     \pgfuseimage{miko@small@dbend}
5533     \xspace
5534 }
5535 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5536 \newrobustcmd\textwarning{
5537   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5538   \xspace
5539 }
5540 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5541 \newrobustcmd\bigtextwarning{
5542   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5543   \xspace
5544 }

```

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

```

5545 \newrobustcmd\putgraphicsat[3]{
5546   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5547 }
5548 \newrobustcmd\putat[2]{
5549   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5550 }

```

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`.

```

5551 \bool_if:NT \c__notesslides_sectocframes_bool {
5552   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5553     \newcounter{chapter}\counterwithin*{section}{chapter}
5554   }{
5555     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5556       \newcounter{chapter}\counterwithin*{section}{chapter}
5557     }
5558   }
5559 }

```

`\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
5560 \def\part@prefix{}
5561 \@ifpackageloaded{document-structure}{}{
5562   \str_case:VnF \__notesslidesstopsect {
5563     {part}{
5564       \int_set:Nn \l_document_structure_section_level_int {0}
5565       \def\thesection{\arabic{chapter}.\arabic{section}}
5566       \def\part@prefix{\arabic{chapter}.}
5567     }
5568     {chapter}{
5569       \int_set:Nn \l_document_structure_section_level_int {1}
5570       \def\thesection{\arabic{chapter}.\arabic{section}}
5571       \def\part@prefix{\arabic{chapter}.}
5572     }
5573   }{
5574     \int_set:Nn \l_document_structure_section_level_int {2}
5575     \def\part@prefix{}

```

```

5576 }
5577 }
5578
5579 \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

```

5580 \renewenvironment{omgroup}[2][]{
5581   \__document_structure_omgroup_args:n { #1 }
5582   \int_incr:N \l_document_structure_omgroup_level_int
5583   \int_incr:N \l_document_structure_section_level_int
5584   \bool_if:NT \c__notesslides_sectocframes_bool {
5585     \stepcounter{slide}
5586     \begin{frame}[noframenumbering]
5587       \vfill\Large\centering
5588       \red{
5589         \ifcase\l_document_structure_section_level_int\or
5590           \stepcounter{part}
5591           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5592           \def\currentsectionlevel{\omdoc@part@kw}
5593         \or
5594           \stepcounter{chapter}
5595           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5596           \def\currentsectionlevel{\omdoc@chapter@kw}
5597         \or
5598           \stepcounter{section}
5599           \def\__notesslideslabel{\part@prefix\arabic{section}}
5600           \def\currentsectionlevel{\omdoc@section@kw}
5601         \or
5602           \stepcounter{subsection}
5603           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5604           \def\currentsectionlevel{\omdoc@subsection@kw}
5605         \or
5606           \stepcounter{subsubsection}
5607           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5608           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5609         \or
5610           \stepcounter{paragraph}
5611           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
5612           \def\currentsectionlevel{\omdoc@paragraph@kw}
5613         \else
5614           \def\__notesslideslabel{}
5615           \def\currentsectionlevel{\omdoc@paragraph@kw}
5616         \fi% end ifcase
5617         \__notesslideslabel%\sref@label@id\__notesslideslabel
5618         \quad #2%
5619       }%
5620     \vfill%
5621     \end{frame}%
5622   }
5623   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5624 }{}
5625 }

```

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

```

5626 \def\inserttheorembodyfont{\normalfont}
5627 %\bool_if:NF \c__notesslides_notes_bool {
5628 % \defbeamertemplate{theorem begin}{miko}
5629 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5630 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5631 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5632 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5633 % \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.

```

5634 % \expandafter\def\csname Parent2\endcsname{}
5635 %}
5636
5637 \AddToHook{begindocument}{% this does not work for some reasons
5638 \setbeamertemplate{theorems}[ams style]
5639 }
5640 \bool_if:NT \c__notesslides_notes_bool {
5641 \renewenvironment{columns}[1][{}]{%
5642 \par\noindent%
5643 \begin{minipage}%
5644 \slidewidth\centering\leavevmode%
5645 }{}%
5646 \end{minipage}\par\noindent%
5647 }%
5648 \newsavebox\columnbox%
5649 \renewenvironment<>{column}[2][{}]{%
5650 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5651 }{}%
5652 \end{minipage}\end{lrbox}\usebox\columnbox%
5653 }%
5654 }
5655 \bool_if:NTF \c__notesslides_noproblems_bool {
5656 \newenvironment{problems}{}{}
5657 }{
5658 \excludecomment{problems}
5659 }

```

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.

```

5660 \gdef\printexcursions{}
5661 \newcommand\excursionref[2]{% label, text
5662 \bool_if:NT \c__notesslides_notes_bool {

```

```

5663 \begin{sparagraph}[title=Excursion]
5664 #2 \sref[fallback=the appendix]{#1}.
5665 \end{sparagraph}
5666 }
5667 }
5668 \newcommand\activate@excursion[2][]{
5669 \gappto\printexcursions{\inputref{#1}{#2}}
5670 }
5671 \newcommand\excursion[4][]{% repos, label, path, text
5672 \bool_if:NT \c__notesslides_notes_bool {
5673 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
5674 }
5675 }

```

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

\excursiongroup

```

5676 \keys_define:nn{notesslides / excursiongroup }{
5677 id .str_set_x:N = \l__notesslides_excursion_id_str,
5678 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5679 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5680 }
5681 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5682 \tl_clear:N \l__notesslides_excursion_intro_tl
5683 \str_clear:N \l__notesslides_excursion_id_str
5684 \str_clear:N \l__notesslides_excursion_mhrepos_str
5685 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5686 }
5687 \newcommand\excursiongroup[1][]{
5688 \__notesslides_excursion_args:n{ #1 }
5689 \ifdefempty\printexcursions{}% only if there are excursions
5690 {\begin{note}
5691 \begin{omgroup}[#1]{Excursions}%
5692 \ifdefempty\l__notesslides_excursion_intro_tl{{
5693 \inputref[\l__notesslides_excursion_mhrepos_str]{
5694 \l__notesslides_excursion_intro_tl
5695 }
5696 }
5697 \printexcursions%
5698 \end{omgroup}
5699 \end{note}}
5700 }
5701 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
5702 \</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.

```
5703 <*package>
5704 <@@=problems>
5705 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5706 \RequirePackage{l3keys2e,expl-keystr-compatible}
5707
5708 \keys_define:nn { problem / pkg }{
5709   notes      .default:n   = { true },
5710   notes      .bool_set:N  = \c__problems_notes_bool,
5711   gnotes     .default:n   = { true },
5712   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
5713   hints      .default:n   = { true },
5714   hints      .bool_set:N  = \c__problems_hints_bool,
5715   solutions  .default:n   = { true },
5716   solutions  .bool_set:N  = \c__problems_solutions_bool,
5717   pts        .default:n   = { true },
5718   pts        .bool_set:N  = \c__problems_pts_bool,
5719   min        .default:n   = { true },
5720   min        .bool_set:N  = \c__problems_min_bool,
5721   boxed      .default:n   = { true },
5722   boxed      .bool_set:N  = \c__problems_boxed_bool,
5723   unknown    .code:n      = {}
5724 }
5725 \newif\ifsolutions
5726
5727 \ProcessKeysOptions{ problem / pkg }
5728 \bool_if:NTF \c__problems_solutions_bool {
5729   \solutionstrue
5730 }{
5731   \solutionsfalse
5732 }
```

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


```
5733 \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.

```
5734 \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.

```
5735 \def\prob@problem@kw{Problem}
5736 \def\prob@solution@kw{Solution}
5737 \def\prob@hint@kw{Hint}
5738 \def\prob@note@kw{Note}
5739 \def\prob@gnote@kw{Grading}
5740 \def\prob@pt@kw{pt}
5741 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5742 \AddToHook{begindocument}{
5743   \ltx@ifpackageloaded{babel}{
5744     \makeatletter
5745     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5746     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5747       \input{problem-ngerman.ldf}
5748     }
5749     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5750       \input{problem-finnish.ldf}
5751     }
5752     \clist_if_in:NnT \l_tmpa_clist {french}{
5753       \input{problem-french.ldf}
5754     }
5755     \clist_if_in:NnT \l_tmpa_clist {russian}{
5756       \input{problem-russian.ldf}
5757     }
5758     \makeatother
5759   }{ }
5760 }
```

40.2 Problems and Solutions

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

```
5761 \keys_define:nn{ problem / problem }{
5762   id      .str_set_x:N = \l__problems_prob_id_str,
5763   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5764   min     .tl_set:N    = \l__problems_prob_min_tl,
5765   title   .tl_set:N    = \l__problems_prob_title_tl,
5766   type    .tl_set:N    = \l__problems_prob_type_tl,
5767   refnum  .int_set:N    = \l__problems_prob_refnum_int
5768 }
5769 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5770 \str_clear:N \l__problems_prob_id_str
5771 \tl_clear:N \l__problems_prob_pts_tl
5772 \tl_clear:N \l__problems_prob_min_tl
5773 \tl_clear:N \l__problems_prob_title_tl
5774 \tl_clear:N \l__problems_prob_type_tl
5775 \int_zero_new:N \l__problems_prob_refnum_int
5776 \keys_set:nn { problem / problem }{ #1 }
5777 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5778   \let\l__problems_prob_refnum_int\undefined
5779 }
5780 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5781 \newcounter{problem}
5782 \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.

```

5783 \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

```

5784 \newcommand\prob@number{
5785   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5786     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5787   }{
5788     \int_if_exist:NTF \l__problems_prob_refnum_int {
5789       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5790     }{
5791       \prob@label\theproblem
5792     }
5793   }
5794 }

```

(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.

```

5795 \newcommand\prob@title[3]{%
5796   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5797     #2 \l__problems_inclprob_title_tl #3
5798   }{
5799     \tl_if_exist:NTF \l__problems_prob_title_tl {
5800       #2 \l__problems_prob_title_tl #3
5801     }{
5802       #1
5803     }
5804   }
5805 }

```

(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.

```
5806 \def\prob@heading{
5807   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5808   %\sref@label{id{\prob@problem@kw~\prob@number}{}}
5809 }
```

(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`

```
5810 \newenvironment{sproblem}[1][{}]{
5811   \__problems_prob_args:n{#1}%\sref@target%
5812   \@in@omtexttrue% we are in a statement (for inline definitions)
5813   \stepcounter{problem}\record@problem
5814   \def\current@section@level{\prob@problem@kw}
5815   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5816     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5817   }{
5818     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5819   }
5820   \str_if_exist:NTF \l__problems_inclprob_id_str {
5821     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5822   }{
5823     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5824   }
5825
5826
5827   \clist_set:No \l_tmpa_clist \sproblemtype
5828   \tl_clear:N \l_tmpa_tl
5829   \clist_map_inline:Nn \l_tmpa_clist {
5830     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5831       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5832     }
5833   }
5834   \tl_if_empty:NTF \l_tmpa_tl {
5835     \__problems_sproblem_start:
5836   }{
5837     \l_tmpa_tl
5838   }
5839   \stex_ref_new_doc_target:n \sproblemid
5840 }{
5841   \clist_set:No \l_tmpa_clist \sproblemtype
5842   \tl_clear:N \l_tmpa_tl
5843   \clist_map_inline:Nn \l_tmpa_clist {
5844     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5845       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5846     }
5847   }
```

```

5847 }
5848 \tl_if_empty:NTF \l_tmpa_tl {
5849   \__problems_sproblem_end:
5850 }{
5851   \l_tmpa_tl
5852 }
5853
5854
5855 \smallskip
5856 }
5857
5858
5859 \cs_new_protected:Nn \__problems_sproblem_start: {
5860   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5861 }
5862 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5863
5864 \newcommand\stexpatchproblem[3][ ] {
5865   \str_set:Nx \l_tmpa_str{ #1 }
5866   \str_if_empty:NTF \l_tmpa_str {
5867     \tl_set:Nn \__problems_sproblem_start: { #2 }
5868     \tl_set:Nn \__problems_sproblem_end: { #3 }
5869   }{
5870     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5871     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5872   }
5873 }
5874
5875
5876 \bool_if:NT \c__problems_boxed_bool {
5877   \surroundwithmdframed{problem}
5878 }

```

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

```

5879 \def\record@problem{
5880   \protected@write\@auxout{}
5881   {
5882     \string\@problem{\prob@number}
5883     {
5884       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5885         \l__problems_inclprob_pts_tl
5886       }{
5887         \l__problems_prob_pts_tl
5888       }
5889     }%
5890     {
5891       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5892         \l__problems_inclprob_min_tl
5893       }{
5894         \l__problems_prob_min_tl
5895       }
5896     }
5897   }
5898 }

```

(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).

```
5899 \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.

```
5900 \keys_define:nn { problem / solution }{
5901   id                .str_set_x:N = \l__problems_solution_id_str ,
5902   for               .tl_set:N    = \l__problems_solution_for_tl ,
5903   height            .dim_set:N   = \l__problems_solution_height_dim ,
5904   creators           .clist_set:N = \l__problems_solution_creators_clist ,
5905   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
5906   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
5907 }
5908 \cs_new_protected:Nn \__problems_solution_args:n {
5909   \str_clear:N \l__problems_solution_id_str
5910   \tl_clear:N \l__problems_solution_for_tl
5911   \tl_clear:N \l__problems_solution_srccite_tl
5912   \clist_clear:N \l__problems_solution_creators_clist
5913   \clist_clear:N \l__problems_solution_contributors_clist
5914   \dim_zero:N \l__problems_solution_height_dim
5915   \keys_set:nn { problem / solution }{ #1 }
5916 }
```

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

```
5917 \newcommand\@startsolution[1][{}]{
5918   \__problems_solution_args:n { #1 }
5919   \@in@omtexttrue% we are in a statement.
5920   \bool_if:NF \c__problems_boxed_bool { \hrule }
5921   \smallskip\noindent
5922   {\textbf\prob@solution@kw : \enspace}
5923   \begin{small}
5924   \def\current@section@level{\prob@solution@kw}
5925   \ignorespacesandpars
5926 }
```

\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.

```
5927 \newcommand\startsolutions{
5928   \specialcomment{solution}{\@startsolution}{
5929     \bool_if:NF \c__problems_boxed_bool {
5930       \hrule\medskip
5931     }
5932     \end{small}%
5933   }
5934   \bool_if:NT \c__problems_boxed_bool {
5935     \surroundwithmdframed{solution}
5936   }
5937 }
```

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

\stopsolutions

```
5938 \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.

```
5939 \ifsolutions
5940 \startsolutions
5941 \else
5942 \stopsolutions
5943 \fi
```

exnote

```
5944 \bool_if:NTF \c__problems_notes_bool {
5945 \newenvironment{exnote}[1][]{
5946 \par\smallskip\hrule\smallskip
5947 \noindent\textbf{\prob@note@kw : }\small
5948 }{
5949 \smallskip\hrule
5950 }
5951 }{
5952 \excludecomment{exnote}
5953 }
```

hint

```
5954 \bool_if:NTF \c__problems_notes_bool {
5955 \newenvironment{hint}[1][]{
5956 \par\smallskip\hrule\smallskip
5957 \noindent\textbf{\prob@hint@kw :~ }\small
5958 }{
5959 \smallskip\hrule
5960 }
5961 \newenvironment{exhint}[1][]{
5962 \par\smallskip\hrule\smallskip
5963 \noindent\textbf{\prob@hint@kw :~ }\small
5964 }{
5965 \smallskip\hrule
5966 }
5967 }{
5968 \excludecomment{hint}
5969 \excludecomment{exhint}
5970 }
```

gnote

```
5971 \bool_if:NTF \c__problems_notes_bool {
5972 \newenvironment{gnote}[1][]{
5973 \par\smallskip\hrule\smallskip
5974 \noindent\textbf{\prob@gnote@kw : }\small
5975 }{
5976 \smallskip\hrule
5977 }
5978 }{
5979 \excludecomment{gnote}
5980 }
```

40.3 Multiple Choice Blocks

```

5981 \newenvironment{mcb}{
5982   \begin{enumerate}
5983 }{
5984   \end{enumerate}
5985 }

```

we define the keys for the mcc macro

```

5986 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5987   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5988     \bool_set_true:N #1
5989   }{
5990     \bool_set_false:N #1
5991   }
5992 }
5993 \keys_define:nn { problem / mcc }{
5994   id          .str_set_x:N = \l__problems_mcc_id_str ,
5995   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5996   T           .default:n   = { true } ,
5997   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5998   F           .default:n   = { true } ,
5999   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6000   Ttext       .code:n      = {
6001     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6002   } ,
6003   Ftext       .code:n      = {
6004     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6005   }
6006 }
6007 \cs_new_protected:Nn \l__problems_mcc_args:n {
6008   \str_clear:N \l__problems_mcc_id_str
6009   \tl_clear:N \l__problems_mcc_feedback_tl
6010   \bool_set_true:N \l__problems_mcc_t_bool
6011   \bool_set_true:N \l__problems_mcc_f_bool
6012   \bool_set_true:N \l__problems_mcc_Ttext_bool
6013   \bool_set_false:N \l__problems_mcc_Ftext_bool
6014   \keys_set:nn { problem / mcc }{ #1 }
6015 }

```

\mcc

```

6016 \newcommand\mcc[2][]{
6017   \l__problems_mcc_args:n{ #1 }
6018   \item #2
6019   \ifsolutions
6020     \\\
6021     \bool_if:NT \l__problems_mcc_t_bool {
6022       % TODO!
6023       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6024     }
6025     \bool_if:NT \l__problems_mcc_f_bool {

```

²³EdNOTE: MK: maybe import something better here from a dedicated MC package

```

6026      % TODO!
6027      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6028    }
6029    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6030      !
6031    }{
6032      \l__problems_mcc_feedback_tl
6033    }
6034    \fi
6035  } %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.

```

6036
6037 \keys_define:nn{ problem / inclproblem }{
6038   id      .str_set:N = \l__problems_inclprob_id_str,
6039   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6040   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6041   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6042   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6043   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6044   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6045 }
6046 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6047   \str_clear:N \l__problems_prob_id_str
6048   \tl_clear:N \l__problems_inclprob_pts_tl
6049   \tl_clear:N \l__problems_inclprob_min_tl
6050   \tl_clear:N \l__problems_inclprob_title_tl
6051   \tl_clear:N \l__problems_inclprob_type_tl
6052   \int_zero_new:N \l__problems_inclprob_refnum_int
6053   \str_clear:N \l__problems_inclprob_mhrepos_str
6054   \keys_set:nn { problem / inclproblem }{ #1 }
6055   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6056     \let\l__problems_inclprob_pts_tl\undefined
6057   }
6058   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6059     \let\l__problems_inclprob_min_tl\undefined
6060   }
6061   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6062     \let\l__problems_inclprob_title_tl\undefined
6063   }
6064   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6065     \let\l__problems_inclprob_type_tl\undefined
6066   }
6067   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6068     \let\l__problems_inclprob_refnum_int\undefined
6069   }
6070 }

```



```

6071
6072 \cs_new_protected:Nn \__problems_inclprob_clear: {
6073   \let\l__problems_inclprob_id_str\undefined
6074   \let\l__problems_inclprob_pts_tl\undefined
6075   \let\l__problems_inclprob_min_tl\undefined
6076   \let\l__problems_inclprob_title_tl\undefined
6077   \let\l__problems_inclprob_type_tl\undefined
6078   \let\l__problems_inclprob_refnum_int\undefined
6079   \let\l__problems_inclprob_mhrepos_str\undefined
6080 }
6081 \__problems_inclprob_clear:
6082
6083 \newcommand\includeproblem[2][ ]{
6084   \__problems_inclprob_args:n{ #1 }
6085   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6086     \input{#2}
6087   }{
6088     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6089       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6090     }
6091   }
6092   \__problems_inclprob_clear:
6093 }

```

(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.

```

6094 \AddToHook{enddocument}{
6095   \bool_if:NT \c__problems_pts_bool {
6096     \message{Total:~\arabic{pts}~points}
6097   }
6098   \bool_if:NT \c__problems_min_bool {
6099     \message{Total:~\arabic{min}~minutes}
6100   }
6101 }

```

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

```

6102 \def\pts#1{
6103   \bool_if:NT \c__problems_pts_bool {
6104     \marginpar{#1~\prob@pt@kw}
6105   }
6106 }
6107 \def\min#1{
6108   \bool_if:NT \c__problems_min_bool {
6109     \marginpar{#1~\prob@min@kw}
6110   }
6111 }

```

`\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.

```

6112 \newcounter{pts}
6113 \def\show@pts{
6114   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6115     \bool_if:NT \c__problems_pts_bool {
6116       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6117       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6118     }
6119   }{
6120     \tl_if_exist:NT \l__problems_prob_pts_tl {
6121       \bool_if:NT \c__problems_pts_bool {
6122         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6123         \addtocounter{pts}{\l__problems_prob_pts_tl}
6124       }
6125     }
6126   }
6127 }

```

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

`\show@min`

```

6128 \newcounter{min}
6129 \def\show@min{
6130   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6131     \bool_if:NT \c__problems_min_bool {
6132       \marginpar{\l__problems_inclprob_min_tl\ min}
6133       \addtocounter{min}{\l__problems_inclprob_min_tl}
6134     }
6135   }{
6136     \tl_if_exist:NT \l__problems_prob_min_tl {
6137       \bool_if:NT \c__problems_min_bool {
6138         \marginpar{\l__problems_prob_min_tl\ min}
6139         \addtocounter{min}{\l__problems_prob_min_tl}
6140       }
6141     }
6142   }
6143 }
6144 \</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.

```
6145 <@@=hwexam>
6146 <*cls>
6147 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6148 \RequirePackage{l3keys2e,expl-keystr-compatible}
6149 \DeclareOption*{
6150   \PassOptionsToClass{\CurrentOption}{document-structure}
6151   \PassOptionsToPackage{\CurrentOption}{stex}
6152   \PassOptionsToPackage{\CurrentOption}{hwexam}
6153   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6154 }
6155 \ProcessOptions
```

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

```
6156 \LoadClass{document-structure}
6157 \RequirePackage{stex}
6158 \RequirePackage{hwexam}
6159 \RequirePackage{tikzinput}
6160 \RequirePackage{graphicx}
6161 \RequirePackage{a4wide}
6162 \RequirePackage{amssymb}
6163 \RequirePackage{amstext}
6164 \RequirePackage{amsmath}
```

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

```

6165 \newcommand\assig@default@type{\hwexam@assignment@kw}
6166 \def\document@hwexamtype{\assig@default@type}
6167 <@@=document_structure>
6168 \keys_define:nn { document-structure / document }{
6169 id .str_set_x:N = \c_document_structure_document_id_str,
6170 hwexamtype .tl_set:N = \document@hwexamtype
6171 }
6172 <@@=hwexam>
6173 </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.

```
6174 \*package>
6175 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6176 \RequirePackage{l3keys2e,expl-keystr-compat}
6177
6178 \newif\iftest\testfalse
6179 \DeclareOption{test}{\testtrue}
6180 \newif\ifmultiple\multiplefalse
6181 \DeclareOption{multiple}{\multipletrue}
6182 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6183 \ProcessOptions
```

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

```
6184 \RequirePackage{keyval}[1997/11/10]
6185 \RequirePackage{problem}
```

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

```
6186 \newcommand\hwexam@assignment@kw{Assignment}
6187 \newcommand\hwexam@given@kw{Given}
6188 \newcommand\hwexam@due@kw{Due}
6189 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6190 blank~for~extra~space}
6191 \def\hwexam@minutes@kw{minutes}
6192 \newcommand\correction@probs@kw{prob.}
6193 \newcommand\correction@pts@kw{total}
6194 \newcommand\correction@reached@kw{reached}
6195 \newcommand\correction@sum@kw{Sum}
6196 \newcommand\correction@grade@kw{grade}
6197 \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

```

6198 \AddToHook{begindocument}{
6199 \ltx@ifpackageloaded{babel}{
6200 \makeatletter
6201 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6202 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6203 \input{hwexam-ngerman.ldf}
6204 }
6205 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6206 \input{hwexam-finnish.ldf}
6207 }
6208 \clist_if_in:NnT \l_tmpa_clist {french}{
6209 \input{hwexam-french.ldf}
6210 }
6211 \clist_if_in:NnT \l_tmpa_clist {russian}{
6212 \input{hwexam-russian.ldf}
6213 }
6214 \makeatother
6215 }{}
6216 }
6217

```

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.

```

6218 \newcounter{assignment}
6219 \numberproblemsin{assignment}
6220 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6221 \keys_define:nn { hwexam / assignment } {
6222 id .str_set:N = \l__hwexam_assign_id_str,
6223 number .int_set:N = \l__hwexam_assign_number_int,
6224 title .tl_set:N = \l__hwexam_assign_title_tl,
6225 type .tl_set:N = \l__hwexam_assign_type_tl,
6226 given .tl_set:N = \l__hwexam_assign_given_tl,
6227 due .tl_set:N = \l__hwexam_assign_due_tl,
6228 loadmodules .code:n = {
6229 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6230 }
6231 }
6232 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6233 \str_clear:N \l__hwexam_assign_id_str
6234 \int_set:Nn \l__hwexam_assign_number_int {-1}
6235 \tl_clear:N \l__hwexam_assign_title_tl
6236 \tl_clear:N \l__hwexam_assign_type_tl
6237 \tl_clear:N \l__hwexam_assign_given_tl
6238 \tl_clear:N \l__hwexam_assign_due_tl
6239 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6240 \keys_set:nn { hwexam / assignment }{ #1 }
6241 }

```

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.

```

6242 \newcommand\given@due[2]{
6243 \bool_lazy_all:nF {
6244 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6245 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6246 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6247 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6248 }{ #1 }
6249
6250 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6251 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6252 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6253 }
6254 }{
6255 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6256 }
6257
6258 \bool_lazy_or:nnF {
6259 \bool_lazy_and_p:nn {
6260 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6261 }{
6262 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6263 }
6264 }{
6265 \bool_lazy_and_p:nn {
6266 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6267 }{
6268 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6269 }
6270 }{ ,~ }
6271
6272 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6273 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6274 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6275 }
6276 }{
6277 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6278 }
6279
6280 \bool_lazy_all:nF {
6281 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6282 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6283 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6284 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6285 }{ #2 }
6286 }

```

`\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.

```

6287 \newcommand\assignment@title[3]{
6288 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6289 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6290 #1
6291 }{
6292 #2\l__hwexam_assign_title_tl#3
6293 }
6294 }{
6295 #2\l__hwexam_inclasssign_title_tl#3
6296 }
6297 }

```

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

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

```

6298 \newcommand\assignment@number{
6299 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6300 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6301 \arabic{assignment}
6302 } {
6303 \int_use:N \l__hwexam_assign_number_int
6304 }
6305 }{
6306 \int_use:N \l__hwexam_inclasssign_number_int
6307 }
6308 }

```

(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.

```

6309 \newenvironment{assignment}[1][ ]{
6310 \__hwexam_assignment_args:n { #1 }
6311 %\sref@target
6312 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6313 \global\stepcounter{assignment}
6314 }{
6315 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6316 }
6317 \setcounter{problem}{0}
6318 \def\current@section@level{\document@hwexamtype}
6319 %\sref@label@id{\document@hwexamtype \thesection}
6320 \begin{@assignment}
6321 }{
6322 \end{@assignment}
6323 }

```


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

```

6324 \def\ass@title{
6325 \protect\document@hwexamtype~\arabic{assignment}
6326 \assignment@title{}\{;\}{} -- \given@due{}\}{}
6327 }
6328 \ifmultiple
6329 \newenvironment{@assignment}{
6330 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6331 \begin{omgroup}[loadmodules]{\ass@title}
6332 }{
6333 \begin{omgroup}{\ass@title}
6334 }
6335 }{
6336 \end{omgroup}
6337 }

```

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

```

6338 \else
6339 \newenvironment{@assignment}{
6340 \begin{center}\bf
6341 \Large@title\strut\
6342 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
6343 \large\given@due{--;\}{}\}{}
6344 \end{center}
6345 }{}
6346 \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.

```

6347 \keys_define:nn { hwexam / inclassignment } {
6348 %id .str_set_x:N = \l__hwexam_assign_id_str,
6349 number .int_set:N = \l__hwexam_inclassign_number_int,
6350 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6351 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6352 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6353 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6354 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6355 }
6356 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6357 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6358 \tl_clear:N \l__hwexam_inclassign_title_tl
6359 \tl_clear:N \l__hwexam_inclassign_type_tl
6360 \tl_clear:N \l__hwexam_inclassign_given_tl
6361 \tl_clear:N \l__hwexam_inclassign_due_tl
6362 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6363 \keys_set:nn { hwexam / inclassignment }{ #1 }
6364 }
6365 \__hwexam_inclassignment_args:n {}
6366
6367 \newcommand\inputassignment[2][{}]{

```

```

6368 \_hwexam_inclassnment_args:n { #1 }
6369 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6370 \input{#2}
6371 }{
6372 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6373 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6374 }
6375 }
6376 \_hwexam_inclassnment_args:n {}
6377 }
6378 \newcommand\includeassignment[2][]{
6379 \newpage
6380 \inputassignment[#1]{#2}
6381 }

```

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

42.4 Typesetting Exams

\quizheading

```

6382 \ExplSyntaxOff
6383 \newcommand\quizheading[1]{%
6384 \def\@tas{#1}%
6385 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6386 \ifx\@tas\@empty\else%
6387 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6388 \fi%
6389 }
6390 \ExplSyntaxOn

```

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

\testheading

```

6391
6392 \def\hwexamheader{\input{hwexam-default.header}}
6393
6394 \def\hwexamminutes{
6395 \tl_if_empty:NTF \testheading@duration {
6396 {\testheading@min}~\hwexam@minutes@kw
6397 }{
6398 \testheading@duration
6399 }
6400 }
6401
6402 \keys_define:nn { hwexam / testheading } {
6403 min .tl_set:N = \testheading@min,
6404 duration .tl_set:N = \testheading@duration,
6405 reqpts .tl_set:N = \testheading@reqpts,
6406 tools .tl_set:N = \testheading@tools
6407 }
6408 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6409 \tl_clear:N \testheading@min
6410 \tl_clear:N \testheading@duration

```

```

6411 \tl_clear:N \testheading@reqpts
6412 \tl_clear:N \testheading@tools
6413 \keys_set:nn { hwexam / testheading }{ #1 }
6414 }
6415 \newenvironment{testheading}[1][]{
6416   \__hwexam_testheading_args:n{ #1 }
6417   \newcount\check@time\check@time=\testheading@min
6418   \advance\check@time by -\theassignment@totalmin
6419   \newif\if@bonuspoints
6420   \tl_if_empty:NTF \testheading@reqpts {
6421     \@bonuspointsfalse
6422   }{
6423     \newcount\bonus@pts
6424     \bonus@pts=\theassignment@totalpts
6425     \advance\bonus@pts by -\testheading@reqpts
6426     \edef\bonus@pts{\the\bonus@pts}
6427     \@bonuspointstrue
6428   }
6429   \edef\check@time{\the\check@time}
6430
6431   \makeatletter\hwexamheader\makeatother
6432 }{
6433   \newpage
6434 }

```

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

\testspace

```

6435 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6436 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6437 \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.

```

6438 <@=problems>
6439 \renewcommand\@problem[3]{
6440   \stepcounter{assignment@probs}
6441   \def\__problemspts{#2}
6442   \ifx\__problemspts\@empty\else
6443     \addtocounter{assignment@totalpts}{#2}
6444   \fi
6445   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6446   \xdef\correction@probs{\correction@probs & #1}%
6447   \xdef\correction@pts{\correction@pts & #2}
6448   \xdef\correction@reached{\correction@reached &}

```

```

6449 }
6450 <@@=hwexam>

```

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

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

```

6451 \newcounter{assignment@probs}
6452 \newcounter{assignment@totalpts}
6453 \newcounter{assignment@totalmin}
6454 \def\correction@probs{\correction@probs@kw}
6455 \def\correction@pts{\correction@pts@kw}
6456 \def\correction@reached{\correction@reached@kw}
6457 \stepcounter{assignment@probs}
6458 \newcommand\correction@table{
6459 \resizebox{\textwidth}{!}{%
6460 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6461 &\multicolumn{\theassignment@probs}{c|}||%|
6462 {\footnotesize\correction@forgrading@kw} &\\ \hline
6463 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6464 \correction@pts & \theassignment@totalpts & \\ \hline
6465 \correction@reached & & \[.7cm]\hline
6466 \end{tabular}}
6467 \end{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}

```