Lean Mathematical Library

The Lean Mathematical Library (mathlib)

Willem Lambooy & Stijn van den Beemt

17 December 2021

Outline

Formal Libraries

- Mechanised proof-checking systems have libraries of formal definitions and proofs.
- Every major system has at least one 'base' library.
- These libraries differ considerably in contents, organisations and purposes.

mathlib is one such library, developed for Lean.

mathlib: A Repository of Mathematical Proofs

- A community-driven effort, hence no central organisation.
- Focus on classical mathematics.
- Inclusion of automation.

Community goal: supporting the formalisation of modern, research-level mathematics.

mathlib as a fork-library of Lean 3

- Leonardo de Moura started the Lean Project in 2013.
- Lean 3 was released in early 2017.
- Much of Lean's core library was factored out after.

This factored-out code became the base for mathlib.

Additional mathlib Facts

- Initially led by Mario Carneiro and Johannes Hölzl at Carnegie Mellon University (Pittsburgh, PA).
- Over two years, mathlib has grown from 15k to 140k LOC.
- 73 contributors and 11 maintainers. More in Section 7.

mathlib is the de facto standard library for both programming and proving in Lean 3.

Lean

- Lean uses a system of dependant types based on CIC.
- It adds two axioms to the type theory:
  - `classical.choice`, and
  - `propext`.
- Quotient types remove the need for setoids.
- The metaprogramming framework enables direct manipulation of the Lean environment.
  - mathlib makes extensive use of this (Section 6).

The Library's Contents

- Lean's core library contains much of the algebraic hierarchy and defines basic datatypes ($\mathbb{N}$, $\mathbb{Z}$, `list`).
- Mostly undergraduate level mathematics at time of the paper (January 2020).

Subdirectory	LOC	Declarations
`data`	41849	10695
`topology`	17382	2702
`tactic`	12184	1679
`algebra`	9830	2794
`analysis`	7962	1237
`order`	6526	1542
`category_theory`	6299	1560
`set_theory`	6163	1394
`measure_theory`	6113	926
`ring_theory`	5683	1080
`linear_algebra`	4511	805
`computability`	4205	575
`group_theory`	4191	1094
`category`	1770	389
`number_theory`	1394	228
`logic`	1195	403
`field_theory`	1002	121
`geometry`	848	70
`meta`	784	135
`algebraic_geometry`	194	29
	140085	34168

Specific Contents

- `algebra`:
  - From `semigroup` to `linear_ordered_field`, `module`, ...
- `data`:
  - Definitions and properties of data structures: $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, sets, multisets, lists, polynomials, vectors, ...
- `topology`:
  - Theories about uniform spaces, metric spaces, and algebraic topological spaces.
  - Definition of the Gromov-Hausdorff space, and proof that it is a Polish space.
- `category_theory`:
  - (Co)limits, monadic adjuctions, monoidal categories, ...
  - Used to describe the Giry monad in measure theory.

More Contents and Applications of Quotients

- `meta` and `tactic` define custom tactics using the metaprogramming framework.
- `computability` includes a proof of the undecidability of the halting problem.
- The library yields structural results about groups, rings and fields: Sylow's theorems, Hilbert's basis theorem, ...

- Quotients are used to define:
  - Multisets (lists up to permutation) and finite sets (multisets without duplicates),
  - Quotient groups, tensor products of modules and colimits of rings,
  - The Stone–Čech compactification and Cauchy completion,
  - Cardinals and ordinals,
  - ...

A Weakness of mathlib

Analysis is a weaker point of mathlib (although `analysis` is one of the largest subdirectories!).

The following have been formalised, however:

- The Fréchet derivative,
- The Bochner integral,
- Basic properties of trigonometric functions.

Classes of Types

- Pioneered in Haskell
- Used in Coq/Isabel
- Lean core:
  - `has_coe`, `decidable`
  - `has_add`, `ring`
- hierarchy, structure and inheritance

Structure Graph

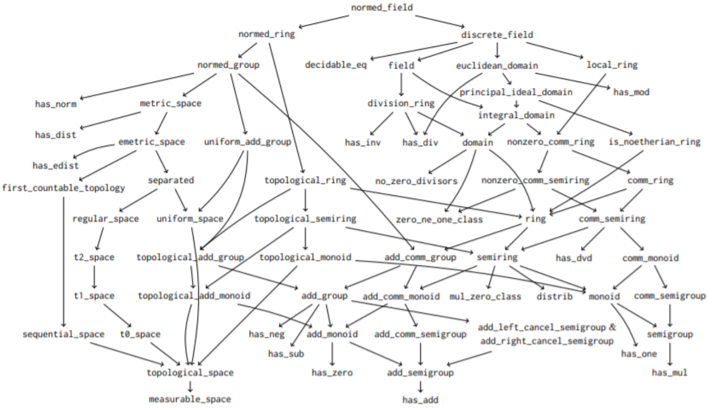

Figure 1. The structure hierarchy underlying `normed_field`. An edge from `S` to `T` indicates that an instance of `T` can be derived from an instance of `S`. Some arrows to the leaves are omitted for readability (e.g. from `zero_ne_one_class` to `has_one`).

(Semi-)Bundled and Unbundled Type Class Definitions

1. Group $= \{(X, ☼) \mid (X, ☼)$ is a group$\}$ (Bundled)
2. group $X = \{☼ \mid (X, ☼)$ is a group$\}$ (Semi-bundled)
3. is_group X ☼ $\iff (X, ☼)$ is a group (Unbundled)

- mathlib primarily utilises semi-bundled type classes (2.).
- Fully bundling is not an option (ambiguity).

How to define (homo)morphisms?

- Bundles (Hom(A, B))?
- Or functions ($f : A \to B$, type class $is\_hom\, f$)?
- mathlib: both
  - composition ($is\_hom\, f \circ g$) hard to infer
  - mostly bundled

decidable p: p $\oplus$ ¬ p

`if p then t else e` is syntactic sugar for `ite p t e`, where:

```
ite : forall (p : Prop) [d : decidable p],
  forall {a : Sort u}, a -> a -> a
```

- (): explicit arguments
- {}: implicit arguments
- [ ]: to be inferred by type class resolution

Decidability

- Propositions shown as (non-computably) decidable using choice axiom, ‘boolean trick’
- *reflect b p*: asserting that the boolean value b is true iff p holds
- decidable p is equivalent to $\sum b$, *reflect b p*

Problems

- mathlib has access to 4256 instances among 386 classes.
- $C\ a \leftarrow C\ a$ can go infinite
  - acyclic instance graph
- Worst case exponential graph search for instances
- Instance search exclusively backward

Together this causes slow search with expansion

- Implemented fixes
- Lean 4
- Working towards fixes

.

Type Classes in Practice

- Small demo
- Show some type class uses
- Not lin-alg course, read the paper

Interactive Exploration of the mathlib Jungles

Metaprogramming in Lean

- The primary automation tool: `simp`.
- Extending the set of rewrite rules through *attributing*:

```
@[simp] lemma mul_zero (m : nat) : mul m 0 = 0 := rfl
```

- Other big automation tools: `finish` and `tidy`.
- Using `#lint` performs linting in the file:
  - Unused arguments,
  - Malformed names,
  - Meeting mathlib's documentation,
  - ...

An example of a special-purpose tactic

- `norm_cast` manipulates type coercions:

Integer inequality: $\uparrow m + \uparrow n > 5$

$\rightarrow$

Natural number inequality: $m + n > 5$

Concluding remarks

- All aforementioned tools (apart from `simp`) are part of mathlib.
- Tools like `norm_cast` are formed on an on-demand basis.

Tactic and tool development as part of the library design.

Community: Contributors

- Fork: Mario Carneiro and Johannes Hölzl.
- LOC: 140k(+?).
- 73 contributors and 11 maintainers at time of article.
- Website says 21 maintainers now, github 191 contributors.
- Mathematics, STEM, to automated proof search to program verification.
- Professors and bachelor students.
- You can help too: GitHub page and Zulip chatroom.

Community: Users and Use Cases

- Two items on a list of Freek and Herman (et al., 2011):
  1. - Unprovability of continuum hypothesis (CH) in "Zermelo-Fraenkel set theory with choice" (ZFC)
     - unprovability of $\neg CH$
     - Possible sizes of infinite sets: $2^{\aleph_0} = \aleph_1$)
  2. - Formalised a proof of "cubing a cube".
     - Any dissection of a cube into smaller cubes must contain at least two cubes of equal width.
- Formalised definition of perfectoid spaces (by Peter Scholze, 2018 Fields Medal). Very complex and thousands of lines.
- Among others (see article), website: 63/100.

Comparison: The best of all worlds?

- Dependent type theory, classical
- Type classes
  - Allows parametrisation of types like $\mathbb{R}^n$
  - Unlike HOLA, HOL Light, Isabelle/HOL
- Though "Many aspects of our structure hierarchy (...) were modeled after similar developments in Isabelle."
- Best comparison: Mathematical Components library, Coq/SSReflect.
  - +: (Almost) fully constructive.
  - -?: More conservative w.r.t (sub)structures.

Conclusions and outlooks

mathlib...

- ...uses dependent type theory
- ...focuses on contemporary mathematics.
- ...provides automation.
- ...hierarchically implements mathematical structures and instances.

In the future and with Lean 4 the contributors hope to improve and grow further. (Lean 4 still experimental)

Further questions?

?