The Lean 3 Mathematical Library (mathlib)

Mario Carneiro

Carnegie Mellon University

July 25, 2018

Lean

- An open source interactive theorem prover developed primarily by Leonardo de Moura (Microsoft Research)
- Focuses on software verification and **formalized mathematics**
- Based on Dependent Type Theory
  - Classical, non-HoTT
- Lean 3 includes a powerful metaprogramming infrastructure for Lean in Lean

A Short History of Lean and mathlib

Several major versions:

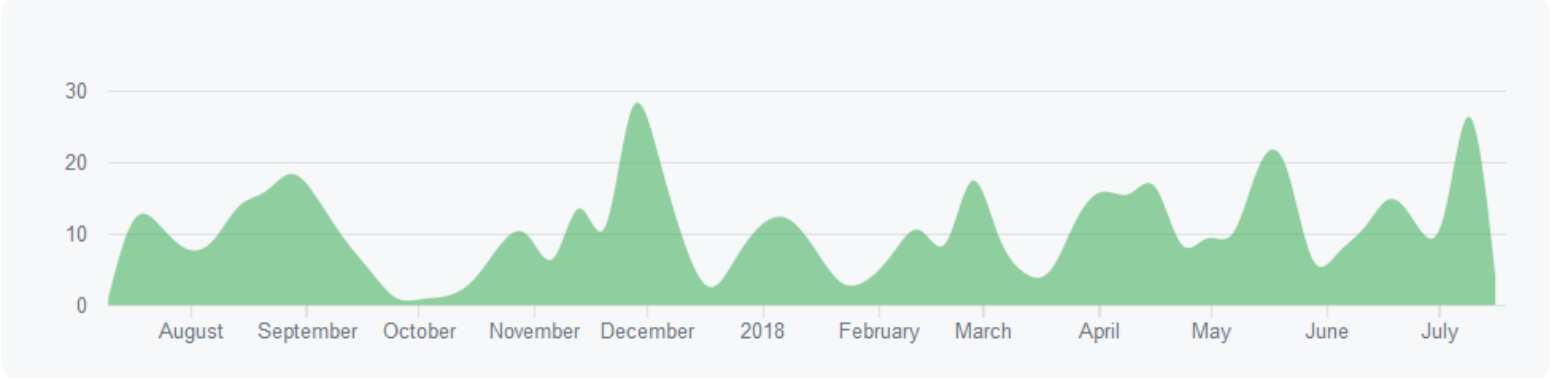
- Lean 1 (no public release)
- Lean 2 (2015) – includes HoTT mode
- Lean 3 (2017)
- Lean 4 in development

- The Lean 2 math library was developed by Jeremy Avigad, Floris van Doorn et al.
- Lean 3 is not backwards compatible with Lean 2, and the decision was made to start again taking advantage of significant new features
- `mathlib` is the latest version of the Lean 3 math library, developed primarily by Mario Carneiro and Johannes Hölzl

Jul 16, 2017 – Jul 25, 2018

Contributions: **Commits**

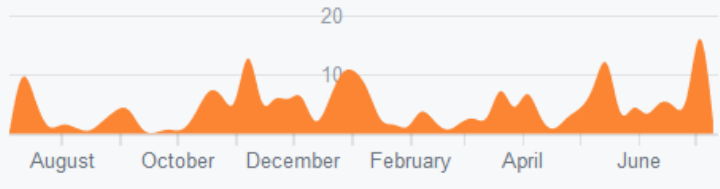
Contributions to master, excluding merge commits



digama0

#1

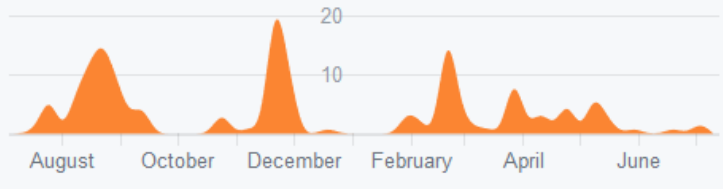
243 commits 95,818 ++ 48,909 --



johoelzl

#2

174 commits 41,608 ++ 23,759 --



Mathlib goals

Two main goals:

- (CS) To build a standard library for lean as a programming language
  - To support verified programming and proven-correct algorithms
  - To support and provide tactics and decision procedures for proof automation
- (Math) To build a library of formalized mathematics, and support users doing the same

These goals complement each other, it is not just two libraries in one

Mathlib vs the core library

- Lean itself has a library, which is even more geared towards CS applications and MS users
- Mathlib is developed on top of this library, and is fully compatible with it, but significantly expands on the mathematics, the (Lean) programming, and the tactics
- The core lean library is currently frozen while Lean 4 is under development, but mathlib is very active

What is in mathlib?

Lean mathematical components library

584 commits | **3** branches | **0** releases | **18** contributors | Apache-2.0

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

cipher1024 and **johoelzl** feat(category/traversable): basic classes for traversable collections — Latest commit `f9cf9d3` 16 days ago

algebra	feat(algebra/pi_instances): more pi instances	a day ago
analysis	refactor(analysis/ennreal): split and move to data.real	4 days ago
category	feat(category/traversable): basic classes for traversable collections	19 hours ago
computability	fix(computability/turing_machine): missed a spot	4 days ago
data	refactor(data/nat/gcd): simplify proof of pow_dvd_pow_iff	3 days ago
docs	doc(wip): finite map (#215) [ci-skip]	a day ago
group_theory	fix(group_theory/group_action): move is_group_action out of namespace	6 days ago
linear_algebra	refactor(data/polynomial): move polynomials to data; replace monomial...	7 days ago
logic	refactor(data/set/countable): define countable in terms of encodable	8 days ago
meta	feat(tactic): add `wlog` (without loss of generality), `tauto`, `auto...	5 months ago
number_theory	refactor(data/set/finite): use hypotheses for fintype assumptions	2 months ago
order	feat(algebra/pi_instances): more pi instances	a day ago
pending	feat(data/real): reals from first principles	6 months ago
ring_theory	fix(*): fix build	7 days ago
set_theory	refactor(data/set/basic): rename set.set_eq_def -> set.ext_iff	8 days ago
tactic	chore(tactic/interactive): change swap so it does what it says	4 days ago
tests	feat(tactic/h_generalize): remove `cast` expressions from goal (#198)	5 days ago
.gitignore	refactor(*): import content from lean/library/data and library_dev	a year ago

Datatypes: `data`

These are all computable where applicable, and so can be used in programming contexts

- $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ – classical number types
- `list` α – finite sequences on α, a.k.a linked lists
- `multiset` α – lists up to permutation (quotient type)
- `finset` α – multisets with no duplicates, used to define finiteness of types and sets
- `array n` α – arrays of fixed length `n` (implemented efficiently in C++)
- `vector` α `n` – lists of fixed length `n` (proven isomorphic to `array n` α)

Datatypes: `data`

- $\alpha \simeq \beta$ (`equiv`) – equivalence/isomorphism of types
- $\alpha \hookrightarrow \beta$ (`embedding`) – injective functions
- `encodable` α – a map from α to $\mathbb{N}$ with partial inverse (Gödel numbering)
- `stream` α, `seq` α, `wseq` α – different kinds of coinductive lists
- `pos_num`, `num`, `znum` – binary natural numbers (for kernel computation)
- `option` α – optional values, nullable types
- `roption` α, `pfun` α β – partial values, functions with precondition (noncomputably isomorphic to `option` α)

Algebraic typeclasses: `algebra`

- `semigroup`, `monoid`, `group`, `semiring`, `ring`, `domain`, `euclidean_domain`, `field` – algebraic structures
- `add_zero : ∀ x, x + 0 = x` – theorems in the first order theory of these structures
- instances showing that the product of groups is a group, the product of rings is a ring, etc

Order structures, sets: `order`, `data.set`

- `preorder`, `partial_order`, `linear_order`, `lattice`, `bounded_lattice`, `complete_lattice`, `conditionally_complete_lattice` – order structures
- `set` α – the collection of all subsets of α, encoded as functions $\alpha \to$ `Prop`
- `filter` α – the collection of all filters on α (which is a complete lattice and a monad)

Topology: `analysis.topology`

- `topological_space` α – a type equipped with an `is_open` predicate
- `nhds a` – the neighborhoods filter
- `map, induced, coinduced` – topological constructions
- `closed, compact, continuous` – topological definitions
- `t1_space, t2_space, regular_space, separable_space, first/second_countable_topology` – topological properties
- `topological_add_group, topological_semiring` – topological algebraic structures

```
instance [t₁ : top α] [t₂ : top β] : top (α × β) :=
induced prod.fst t₁ ⊔ induced prod.snd t₂
```

Theories

- `uniform_space`, `metric_space`
- `measure_theory` – measure spaces, measurable functions, outer measures, measures, Lebesgue measure
- `group_theory` – group actions, subgroups, quotient groups
- `ring_theory` – ideals and local rings
  - much more work is currently happening in the community but not yet in mathlib; Kevin Buzzard is working on schemes and perfectoid spaces
- `computability` – primitive and partial recursive functions, Turing machines, universality and the halting problem
- `number_theory` – the Pell equation, Diophantine equations
- `set_theory` – cardinal and ordinal numbers, computable ordinal notations, large cardinals, a model of ZFC

Tactics: `tactic.interactive`

Mostly minor improvements to the lean core tactic library

- `rcases`, `rintro` – cases with a pattern
- `finish` (Jeremy Avigad), `tauto` (Simon Hudon) – general purpose automation
- `norm_num` – decision procedure for numeric calculation
- `ring`, `ring2` – decision procedure for rings

There are some more tactics available in the community:

- `super` – superposition prover (Gabriel Ebner)
- `cooper` – decision procedure for Presburger arithmetic (Seul Baek)

`docs/tactics.md` has a more complete listing

Future work

- Everything!
- Need more basic analysis – derivatives and integrals
- Almost no number theory except what was needed for MDRP
- Geometry and trigonometry completely absent
- Many basic data structures are missing (binary search trees, association lists) because some techniques like memoized thunks are waiting for lean 4

Thank you!

`https://github.com/leanprover/mathlib`