Maintaining a Library of Formal Mathematics

Floris van Doorn[1], Gabriel Ebner[2], Robert Y. Lewis[2]

[1]University of Pittsburgh

[2]Vrije Universiteit Amsterdam

CICM 2020
2020-07-28

mathlib

- Mathematical library
- Lean proof assistant
- de-facto standard library

- Dependent type theory
- Classical logic (and choice)

- Active community:
  - **github**.com/leanprover-community/mathlib
  - leanprover.**zulip**chat.com

Contents

⋮

Topological algebra order topology, intermediate value theorem, extreme value theorem, limit infimum and supremum, topological group, completion of an abelian topological group, infinite sum, topological ring, completion of a topological ring, topological module.

Metric spaces metric space, ball, sequential compactness is equivalent to compactness (Bolzano-Weierstrass), Heine-Borel theorem (proper metric space version), Lipschitz functions, contraction mapping theorem, Baire theorem, Arzela-Ascoli theorem, Hausdorff distance, Gromov-Hausdorff space.

See also our documentation page about topology.

Analysis

Normed vector spaces normed vector space over a normed field, topology on a normed vector space, equivalence of norms in finite dimension, finite dimensional normed spaces over complete normed fields are complete, Heine-Borel theorem (finite dimensional normed spaces are proper), continuous linear maps, norm of a continuous linear map, Banach open mapping theorem, absolutely convergent series in Banach spaces, Hahn-Banach theorem, completeness of spaces of bounded continuous normed-space-valued functions.

Differentiability differentiable functions between normed vector spaces, derivative of a composite function, derivative of a reciprocal function, Rolle's theorem, mean value theorem, C^k functions, Leibniz formula, local extrema, inverse function theorem, implicit function theorem, analytic function.

Convexity convex functions, characterization of convexity, convexity inequalities, Carathéodory's theorem.

Special functions logarithms, exponential, circular trigonometric functions, hyperbolic trigonometric functions.

Measures and integral calculus sigma-algebras, measurable functions, the category of measurable space, Borel sigma-algebras, positive measure, Lebesgue measure, Giry monad, integral of positive measurable functions, vector-valued integrable functions (Bochner integral), monotone convergence theorem, Fatou's lemma, dominated convergence theorem.

Differentiable manifolds smooth manifold (with boundary and corners), tangent bundle, tangent map.

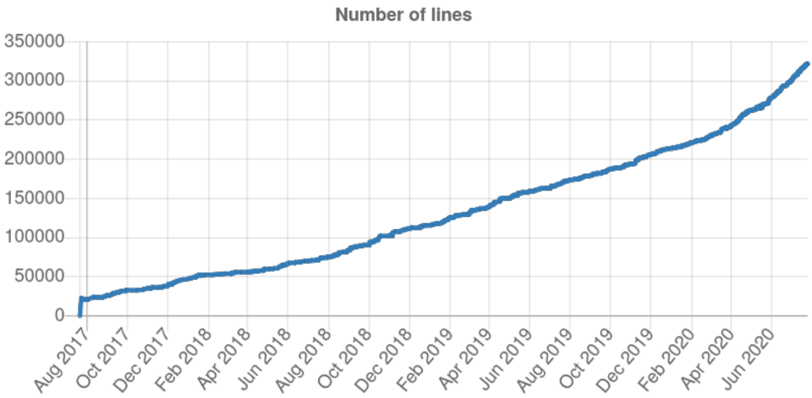
Logic and computation

Computability computable function, Turing machine, halting problem, Rice theorem, combinatorial game.

Set theory ordinal, cardinal, model of ZFC.

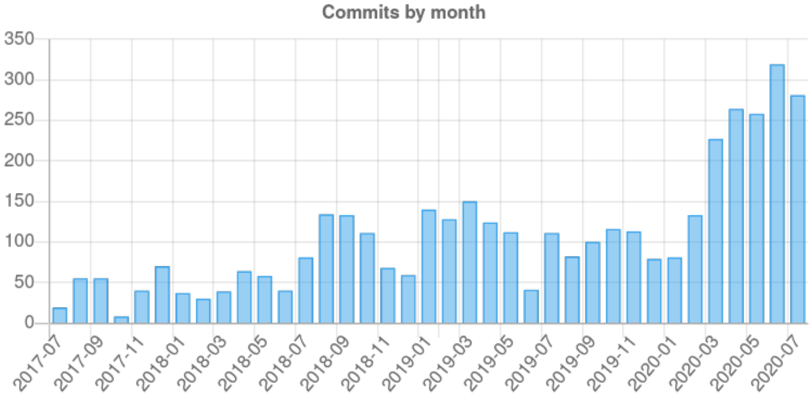
https://leanprover-community.github.io/mathlib-overview.html

Rapid growth



`https://leanprover-community.github.io/mathlib_stats.html`

Rapid growth



https://leanprover-community.github.io/mathlib_stats.html

mathlib contributions

- Around 10 pull requests per day
- 44 active contributors this month
- Many contributors are research mathematicians
- 15 maintainers

→ Simplify reviewing process
→ Promote documentation
→ Establish code quality standards

Interventions

- Documentation website
- Linters
- Continuous integration

Semantic linting

Documentation

Conclusion

Semantic linters

- Implemented as Lean metaprogram
- Inspect each elaborated declaration:

```
meta structure linter :=
(test : declaration → tactic (option string))
-- not shown: header for the error message, etc.
```

- Can be run manually using `#lint`

Unused arguments

- Detects unused arguments in definitions
- Detects unused hypotheses in proofs

```
lemma ex (x y : ℕ) :
  2*x + y < 10 → 5 < y → x < 5 :=
by intros; linarith

/- UNUSED ARGUMENTS: -/
#print ex /- argument 4: (a : 5 < y) -/
```

Maintaining with linters

- Linters enable large-scale refactoring.

```
theorem one_div_pow (ha : a ≠ 0) (n : ℕ) :
  (1 / a) ^ n = 1 / a ^ n :=
...
```

Maintaining with linters

- Linters enable large-scale refactoring.
- March 2020: we add $0^{-1} = 0$ as a field axiom.

```
theorem one_div_pow (ha : a ≠ 0) (n : ℕ) :
  (1 / a) ^ n = 1 / a ^ n :=
...

/- UNUSED ARGUMENTS: -/
#print one_div_pow /- argument 3: (ha : a ≠ 0) -/
```

Maintaining with linters

- Linters enable large-scale refactoring.
- March 2020: we add $0^{-1} = 0$ as a field axiom.

```
theorem one_div_pow                 (n : ℕ) :
  (1 / a) ^ n = 1 / a ^ n :=
...
```

- Preserves code quality over time.

nolint

- Can be disabled if necessary:

```
/--
Constant function.
Ignores the second argument.
-/
@[nolint unused_arguments]
def const (a : α) : β → α :=
λ b, a
```

- Errors from before linters were introduced are grandfathered in
  - `nolints.txt` contains 1724 exceptions

Simplifier

- `simp` tactic does (conditional) term rewriting

```
example (x : ℂ) :
  deriv (λ x, cos (sin x) * exp x) x =
    -(sin (sin x) * cos x * exp x)
      + cos (sin x) * exp x :=
by simp
```

- >7000 lemmas marked as `@[simp]`

Simplifier

```
@[simp] lemma zero_add : 0 + x = x := ...
@[simp] lemma add_zero : x + 0 = x := ...
@[simp] lemma mul_one  : x * 1 = x := ...

example : 0 + (x * 1 + 0) = x := by simp
```

- Terms are rewritten inside-out until in normal form

$$\mathtt{0 + (\underline{x * 1} + 0)} \rightsquigarrow \mathtt{0 + (\underline{x + 0})} \rightsquigarrow \mathtt{\underline{0 + x}} \rightsquigarrow \mathtt{x}$$

Simplifier

```
@[simp] lemma zero_add : 0 + x = x := ...
@[simp] lemma add_zero : x + 0 = x := ...
@[simp] lemma mul_one  : x * 1 = x := ...

example : 0 + (x * 1 + 0) = x := by simp
```

- Terms are rewritten inside-out until in normal form

$$0 + (\underline{x * 1} + 0) \rightsquigarrow 0 + (\underline{x + 0}) \rightsquigarrow \underline{0 + x} \rightsquigarrow x$$

- Lemmas whose left-hand side is not in normal form are never used:

```
@[simp] lemma never_used : x * (0 + x) = x^2 := ...
```

simp_nf linter for redundant simp lemmas

```
@[simp] lemma never_used : x * (0 + x) = x^2 :=
by simp [nat.pow_succ]

/- SOME SIMP LEMMAS ARE NOT IN SIMP-NORMAL FORM.
see note [simp-normal form] for tips how to debug
this.
: -/
#print never_used /- Left-hand side simplifies from
  x * (0 + x)
to
  x * x
using
  [zero_add]
Try to change the left-hand side to the simplified
term!
 -/
```

`simp_nf` linter for redundant simp lemmas

- >100 redundant lemmas found at introduction
- Often due to interaction between multiple files

  "I've already learnt something new about `simp` by looking at my first linter output." Kevin Buzzard

Other linters

- Definitions without documentation
- Uses `def` instead of `lemma`
- Uses $\geq$ instead of $\leq$
- Uses `decidable` instead of choice in theorems
- Type-class instances that can cause loops
- Type-class instances that are slow
- ...
- Full list at: `https://leanprover-community.github.io/mathlib_docs/commands.html#linting%20commands`

Semantic linting

Documentation

Conclusion

Kinds of documentation

- Directly next to the thing (internal documentation)
  - Module docstrings
  - Declaration docstrings
- Scattered over mathlib (decentralized documentation)
  - Library notes
  - Tactic descriptions
- Website (`leanprover-community.github.io`)
  - Style guide
  - High-level overview guides
  - ...

Module docstring

```
/-!
# Germ of a function at a filter

The germ of a function `f : α → β` at a filter
`l : filter α` is the equivalence class of `f`
with respect to the equivalence relation
`eventually_eq l`:
`f ≈ g` means `∀ᶠ x in l, f x = g x`.

## Main definitions

We define
* `germ l β` to be the space of germs of functions
  `α → β` at a filter `l : filter α`;
* coercion from `α → β` to `germ l β`: `(f : germ l β)`
  is the germ of `f : α → β` at `l : filter α`; ...
```

- At the beginning of each file
- Mandatory

Declaration docstring

```
/--
Let `f` be a function between two smooth manifolds.
Then `mfderiv I I′ f x` is the derivative of `f` at `x`,
as a continuous linear map from the tangent space
at `x` to the tangent space at `f x`.
-/
def mfderiv (f : M → M′) (x : M) :
  tangent_space I x →L[𝕜] tangent_space I′ (f x) :=
if h : mdifferentiable_at I I′ f x then
(fderiv_within 𝕜
  (written_in_ext_chart_at I I′ x f : E → E′)
  (range I) ((ext_chart_at I x) x) : _)
else 0
```

- Mandatory for definitions

Declaration docstring

```
def mfderiv {…}                                          view source
        (I : model_with_corners 𝕜 E H)
        (I' : model_with_corners 𝕜 E' H') (f : M → M')
        (x : M) :
    tangent_space I x →L[𝕜] tangent_space I' (f x)
```

Let `f` be a function between two smooth manifolds. Then `mfderiv I I' f x` is the derivative of `f` at `x`, as a continuous linear map from the tangent space at `x` to the tangent space at `f x`.

▸ Equations

- `→L[]`, etc., links to definition
- `{...}` expands to show implicit arguments

Library notes

- Inspired by a technique used in GHC (Glasgow Haskell Compiler)
- Documentation for common design decisions
- `group_theory/coset.lean`:

```
/--
We use the class `has_coe_t` instead of `has_coe`
if the first argument ...
-/
library_note "use has_coe_t"
```

- `topology/uniform_space/completion.lean`:

```
instance : has_coe_t α (completion α) :=
⟨quotient.mk ∘ pure_cauchy⟩
-- note [use has_coe_t]
```

Tactic descriptions

- Documents family of tactics

```
/--
The `norm_cast` family of tactics is used to normalize
casts inside expressions.
... [long description] ...
-/
add_tactic_doc
{ name := "norm_cast",
  category    := doc_category.tactic,
  decl_names := [``norm_cast, ``rw_mod_cast,
    ``apply_mod_cast, ``assumption_mod_cast,
    ``exact_mod_cast, ``push_cast],
  tags        := ["coercions", "simplification"] }
```

Semantic linting

Documentation

Conclusion

Conclusion

- Positive reception among users
- Noticeable improvements
  - Documentation
  - simp lemmas
  - Type class performance
- Reviewers can focus on technical content