Introduction to Lean

Jake Levinson
Simon Fraser University
Computational Math Day

May 4, 2023

What is Lean?

- Lean is an **interactive proof assistant**: you type in a proof and it verifies it

```
example (p q r : Prop) :
  (p → q) → (q → r) → p → r :=
begin
  intros hpq hqr hp,
  apply hqr,
  apply hpq,
  exact hp,
end
```

```
▼ Tactic state

1 goal

p q r : Prop
hpq : p → q
hqr : q → r
hp : p
⊢ r
```

```
▼ jl2023.lean:127:0

type mismatch, term
  hqp hq
has type
  p
but is expected to have type
  r
```

What is Lean?

- Lean is an **interactive proof assistant**: you type in a proof and it verifies it

```
example (p q r : Prop) :
  (p → q) → (q → r) → p → r :=
begin
  intros hpq hqr hp,
  apply hqr,
  apply hpq,
  exact hp,
end
```

```
▼Tactic state

1 goal

p q r : Prop
hpq : p → q
hqr : q → r
hp : p
⊢ r
```

```
▼jl2023.lean:127:0

type mismatch, term
  hqp hq
has type
  p
but is expected to have type
  r
```

- Lean is **tactic-based**: it has some limited (but essential) ability to fill in boring details of proofs

```
example (x y : ℝ) :
  (x - y) * (x + y) = x^2 - y^2 :=
begin
  ring,
end
```

```
▼Tactic state

goals accomplished 🎉

▶All Messages (0)
```

Why formalize?

Objectives of formalization:

- **Verify correctness** of theorems
- **Generate** proofs automatically (especially boring, rote computations)
- **State results** precisely (and look them up)

Formalization has a long history prior to Lean (Coq, Isabelle, ...).

mathlib: the mathematics library

- Lean is a **strictly-typed programming language**, designed by Leonardo de Moura (Microsoft Research)
- Open-source mathematics library: **mathlib**
  https://github.com/leanprover-community/mathlib/
  - 1M+ lines of code, covering most of undergrad math, lots of grad math, some research-level math

mathlib: the mathematics library

- Lean is a **strictly-typed programming language**, designed by Leonardo de Moura (Microsoft Research)
- Open-source mathematics library: **mathlib**
  https://github.com/leanprover-community/mathlib/
  - 1M+ lines of code, covering most of undergrad math, lots of grad math, some research-level math
  - Overview of mathlib: here. Check out: intermediate value theorem, implicit function theorem, insolvability of the quintic, Haar measure, Hilbert's nullstellensatz ...
  - Liquid Tensor Experiment (Commelin et al. 2022) : a lemma on perfectoid spaces, proposed as a challenge by Fields medalist Peter Scholze and his collaborator Dustin Clausen
- Discussion forum: https://leanprover.zulipchat.com/
  Very active and responsive on the new members channel!

Quick primer on type theory

Every object in Lean has a *type*:

`object`	:	`Type`	"`object` is of the stated type"
n	:	$\mathbb{N}$	n is a **natural number**
$\sin$	:	$\mathbb{R} \to \mathbb{R}$	$\sin$ is a **function from** $\mathbb{R}$ **to** $\mathbb{R}$
$x > 0$	:	Prop	"$x > 0$" is a **proposition**
h	:	$x > 0$	h is a **proof** of the proposition $x > 0$
$\mathbb{R}, \mathrm{Prop}$	:	Type	"real" and "proposition" are **Types**

Quick primer on type theory

Every object in Lean has a *type*:

`object`	:	`Type`	"`object` is of the stated type"
n	:	$\mathbb{N}$	n is a **natural number**
sin	:	$\mathbb{R} \to \mathbb{R}$	sin is a **function from** $\mathbb{R}$ **to** $\mathbb{R}$
$x > 0$	:	Prop	"$x > 0$" is a **proposition**
h	:	$x > 0$	h is a **proof** of the proposition $x > 0$
$\mathbb{R}, \mathrm{Prop}$	:	Type	"real" and "proposition" are **Types**

Some examples:

```
def x : ℝ := 5
```

x is the real number 5

```
def seq_limit (a : ℕ → ℝ) (l : ℝ) : Prop :=
∀ ε > 0, ∃ N, ∀ n ≥ N, |a n - l| < ε
```

"$\lim a_n = \ell$" is defined as the proposition that $\forall \epsilon > 0, \ldots$

```
lemma fermat :
  ∀ (a b c n : ℕ) (hn : n > 2),
  a*b*c ≠ 0 → a^n + b^n ≠ c^n := sorry
```

Fermat's proof that $a^n + b^n \neq c^n$ for $n > 2$ goes as follows: *(omitted for lack of space)*

Trying this out!

Let's do some basics first.

Learning resources

- The Natural Number Game: all about `induction`!
  https://www.ma.imperial.ac.uk/~buzzard/xena/natural_number_game/
  Runs in the browser – easiest to get started!

- Patrick Massot's Lean `tutorial`: basic real analysis, culminating in the Intermediate Value Theorem:
  Run `leanproject get tutorials` (command line) or follow download instructions at
  https://github.com/leanprover-community/tutorials
  Start with the file `src/exercises/01_equality_rewriting.lean`.

- Exercises from Lean for the Curious Mathematician 2020: broader overview, organized by topic (analysis, algebra, etc).
  https://github.com/leanprover-community/lftcm2020 or
  `leanproject get lftcm2020` (command line)

Trouble installing Lean?

- You can use Gitpod gitpod.io to run Lean and other mathlib-based projects in a browser.
  You get 10 hours a month for free.
- The Lean Zulip chat is very friendly and very helpful!
  `https://leanprover.zulipchat.com/`