Lean 4 the curious statisticians

Arshak Parsa

Where does the name come from?

Lean for the Curious Mathematician 2022

Disclaimer

- I am currently learning Lean4, so there might be some mistakes in this representation.

Objectives

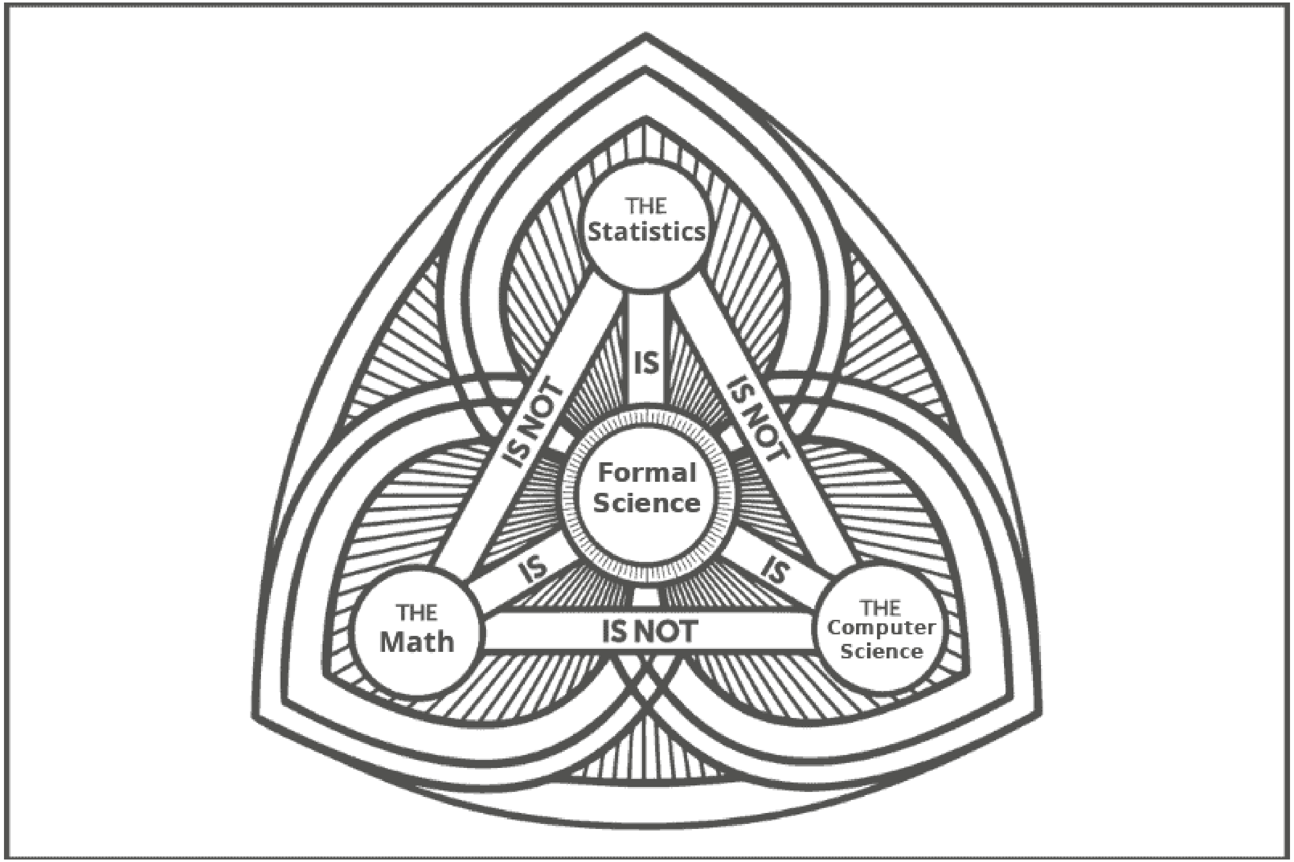
- I want to introduce Lean4 to you.
- I also introduce type theory.

Glossary

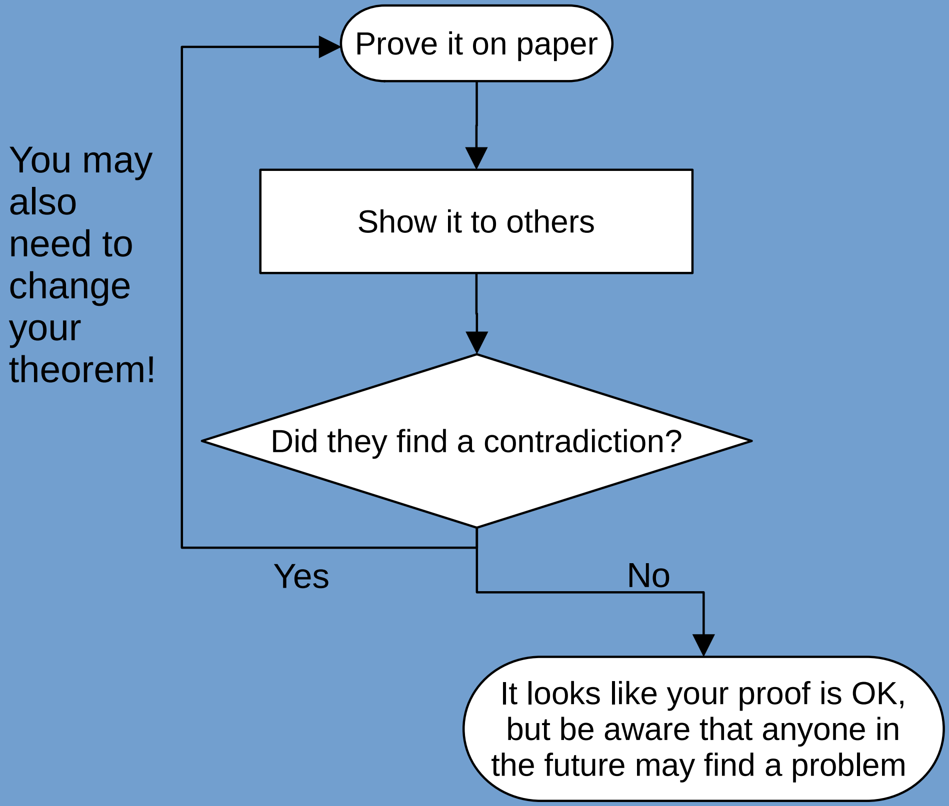
- Proof Assistant : A piece of software that can verify correctness of your theorems, e.g. Lean4.
- Formalize : To type a theorem into a computer proof assistant.
- Type theory : Type theory can be used as a foundation for mathematics. (Lean4 is based on type theory)

Why do we need a proof assistant?

Remember the Holy Trinity

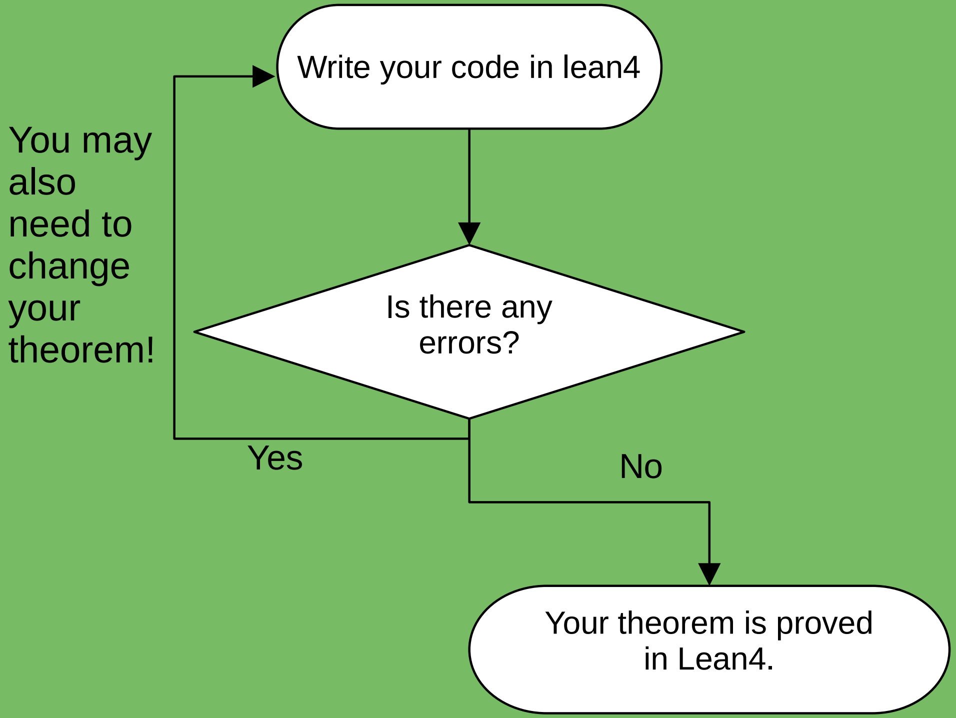


How we were used to prove



You need pen, paper and other people

What is new?



You need a computer with lean4 installed

How a theorem used to look like

Consider the probability space $(\Omega, \mathcal{F}, P)$. Suppose A is measurable, then:

$$1 - P(A) = P(A^{\mathsf{c}})$$

$$\begin{aligned} 1 - P(A) &= P(\Omega) - P(A) \\ &= P(A \cup A^{\mathsf{c}}) - P(A) \\ &= P(A) + P(A^{\mathsf{c}}) - P(A) \\ &= P(A^{\mathsf{c}}) \end{aligned}$$

You use latex to represent your theorem

What is new?

```
theorem P_compl
  {Ω : Type} [MeasurableSpace Ω] {P : Measure Ω}
  [IsProbabilityMeasure P] {A : Set Ω} (hm : MeasurableSet A):
    P (Aᶜ) = 1 - P (A) := by
  apply Eq.symm
  calc
    1 - P (A) = P (univ) - P (A) := by
      rw [measure_univ]
    _ = P (A ∪ Aᶜ) - P (A) := by
      simp
    _ = P (A) + P (Aᶜ) - P (A) := by
      rw [measure_union' disjoint_compl_right hm]
    _ = P (Aᶜ) := by
      simp [measure_ne_top]
```

You may use calc mode.

How a theorem used to look like

Consider the probability space $(\Omega, \mathcal{F}, P)$. Suppose A is measurable, then:

$$1 - P(A) = P(A^c)$$

$$\begin{aligned} 1 - P(A) &= P(\Omega) - P(A) \\ &= P(A \cup A^c) - P(A) \\ &= P(A) + P(A^c) - P(A) \\ &= P(A^c) \end{aligned}$$

You use latex to represent your theorem

What is new?

```
theorem P_compl
  {Ω : Type} [MeasurableSpace Ω] {P : Measure Ω}
  [IsProbabilityMeasure P] {A : Set Ω} (hm : MeasurableSet A):
    P (Aᶜ) = 1 - P (A) := by
  apply Eq.symm
  calc
    1 - P (A) = P (univ) - P (A) := by
      rw [measure_univ]
    _ = P (A ∪ Aᶜ) - P (A) := by
      simp
    _ = P (A) + P (Aᶜ) - P (A) := by
      rw [measure_union' disjoint_compl_right hm]
    _ = P (Aᶜ) := by
      simp [measure_ne_top]
```

You may use calc mode.

How a theorem used to look like

Consider the probability space $(\Omega, \mathcal{F}, P)$. Suppose A is measurable, then:

$$1 - P(A) = P(A^{\mathsf{c}})$$

$$\begin{aligned} 1 - P(A) &= P(\Omega) - P(A) \\ &= P(A \cup A^{\mathsf{c}}) - P(A) \\ &= P(A) + P(A^{\mathsf{c}}) - P(A) \\ &= P(A^{\mathsf{c}}) \end{aligned}$$

You use latex to represent your theorem

What is new?

```
theorem P_compl
  {Ω : Type} [MeasurableSpace Ω] {P : Measure Ω}
  [IsProbabilityMeasure P] {A : Set Ω} (hm : MeasurableSet A):
  P (Aᶜ) = 1 - P (A) := by
  apply Eq.symm
  calc
    1 - P (A) = P (univ) - P (A) := by
      rw [measure_univ]
    _ = P (A ∪ Aᶜ) - P (A) := by
      simp
    _ = P (A) + P (Aᶜ) - P (A) := by
      rw [measure_union' disjoint_compl_right hm]
    _ = P (Aᶜ) := by
      simp [measure_ne_top]
```

You may use calc mode.

How a theorem used to look like

Consider the probability space $(\Omega, \mathcal{F}, P)$. Suppose A is measurable, then:

$$\boxed{1 - P(A) = P(A^{c})}$$

$$\begin{aligned}
1 - P(A) &= P(\Omega) - P(A) \\
&= P(A \cup A^{c}) - P(A) \\
&= P(A) + P(A^{c}) - P(A) \\
&= P(A^{c})
\end{aligned}$$

You use latex to represent your theorem

What is new?

```
theorem P_compl
  {Ω : Type} [MeasurableSpace Ω] {P : Measure Ω}
  [IsProbabilityMeasure P] {A : Set Ω} (hm : MeasurableSet A):
    P (Aᶜ) = 1 - P (A) := by
  apply Eq.symm
  calc
    1 - P (A) = P (univ) - P (A) := by
      rw [measure_univ]
    _ = P (A ∪ Aᶜ) - P (A) := by
      simp
    _ = P (A) + P (Aᶜ) - P (A) := by
      rw [measure_union' disjoint_compl_right hm]
    _ = P (Aᶜ) := by
      simp [measure_ne_top]
```

You may use calc mode.

How a theorem used to look like

Consider the probability space $(\Omega, \mathcal{F}, P)$. Suppose A is measurable, then:

$$1 - P(A) = P(A^{\complement})$$

$$\boxed{\begin{aligned} 1 - P(A) &= P(\Omega) - P(A) \\ &= P(A \cup A^{\complement}) - P(A) \\ &= P(A) + P(A^{\complement}) - P(A) \\ &= P(A^{\complement}) \end{aligned}}$$

You use latex to represent your theorem

What is new?

```
theorem P_compl
  {Ω : Type} [MeasurableSpace Ω] {P : Measure Ω}
  [IsProbabilityMeasure P] {A : Set Ω} (hm : MeasurableSet A):
    P (Aᶜ) = 1 - P (A) := by
  apply Eq.symm
  calc
    1 - P (A) = P (univ) - P (A) := by
      rw [measure_univ]
    _ = P (A ∪ Aᶜ) - P (A) := by
      simp
    _ = P (A) + P (Aᶜ) - P (A) := by
      rw [measure_union' disjoint_compl_right hm]
    _ = P (Aᶜ) := by
      simp [measure_ne_top]
```

You may use calc mode.

How a theorem used to look like

What is new?

5. $P(\tilde{A}) = 1 - P(A)$ for every $A \subset \Omega$.

```
theorem P_compl [d : DiscreteDistFunc Ω] (A : Set Ω):
  P (Aᶜ) = 1 - P (A) := by
  rewrite [← (@P_eq_one Ω d)]
  rewrite [(Set.compl_union_self A).symm]
  rewrite [P_union_disjoint]
  simp
  exact Set.disjoint_compl_left_iff_subset.mpr fun ⦃a⦄ a ↦ a
```

Finally, to prove Property 5, consider the disjoint union

$$\Omega = A \cup \tilde{A} \ .$$

Since $P(\Omega) = 1$, the property of disjoint additivity (Property 4) implies that

$$1 = P(A) + P(\tilde{A}) \ ,$$

whence $P(\tilde{A}) = 1 - P(A)$. $\square$

You use latex to represent your theorem

You may use blueprint to link latex to lean4

How a theorem used to look like	What is new?
A theorem is a statement that has been proven.	A theorem is a function! You give some input, you get a proof!

```
theorem t1
  (n : ℕ) (m : ℕ) :
    n + m = m + n :=
  Nat.add_comm n m

-- t1 is a function

theorem t2 :
    1 + 4 = 4 + 1 :=
  t1 1 4
```

If you can’t do it better, doesn’t mean it’s impossible!

```lean
theorem iUnion_eq_iUnion_diff
  {A : ℕ → Set Ω} (hm : Monotone A) (h0 : A 0 = ∅) :
    (⋃ i, A i) = ⋃ i, A i \ A (i - 1) := by
  ext x
  constructor
  · intro h
    let Nx := {n : ℕ | x ∈ (A n)}
    apply exists_exists_eq_and.mp at h
    obtain (k, h) := h
    have hNx : (Nx).Nonempty := by
      exact Set.nonempty_of_mem h
    let Nx_min := WellFounded.min wellFounded_lt Nx hNx
    have hn : ∀m ∈ Nx, Nx_min≤m := by
      exact fun m a => WellFounded.min_le wellFounded_lt a hNx
    have hNx_min_mem : Nx_min ∈ Nx := by
      exact WellFounded.min_mem wellFounded_lt Nx hNx
    have hz : Nx_min≠0 := by
      aesop
    match Nx_min with
    | 0 => exact False.elim (hz rfl)
    | 1 =>
      aesop
    | k1+2 =>
      set k := k1+2
      have hkm1 : (k-1) ∉ Nx := by
        by_contra hf
        have h1 : k-1 ≤ k → k = k-1 := by
          exact fun a => Nat.le_antisymm (hn (k - 1) (hm (hn (k - 1) hf) hNx_min_mem)) a
        aesop
      have h3 : x ∈ A k := by
        exact hm (hn k hNx_min_mem) hNx_min_mem
      aesop
  · intro h
    simp_all only [Set.mem_iUnion, Set.mem_diff]
    obtain (w, h) := h
    obtain (left, _) := h
    apply Exists.intro
    · exact left
```

```lean
theorem iUnion_eq_iUnion_diff2
  {A : ℕ → Set Ω} (hm : Monotone A) (h0 : A 0 = ∅) :
    (⋃ i, A i) = ⋃ i, A i \ A (i - 1) := by
  rw [← iUnion_disjointed]
  rw [← Set.union_iUnion_nat_succ]
  simp_rw [Monotone.disjointed_succ hm]
  apply Eq.symm
  rw [← Set.union_iUnion_nat_succ]
  simp [h0]
```

new members > Increasing Sequence of Sets induces Partition on Limit — SEP 17

Arshak Parsa 1:56 PM

Can someone prove this?
(Is it even provable?)

```lean
import Mathlib

theorem iUnion_eq_iUnion_diff
  {A : ℕ → Set Ω} (hm : Monotone A) (h0 : A 0 = ∅) :
    (⋃ i, A i) = ⋃ i, A i \ A (i - 1) := by
  ext x
  constructor
  · intro h
    simp_all only [Set.mem_iUnion, Set.mem_diff]
    obtain (w, h) := h
    sorry
  · intro h
    simp_all only [Set.mem_iUnion, Set.mem_diff]
    obtain (w, h) := h
    obtain (left, right) := h
    apply Exists.intro
    · exact left
```

I found a proof here but I don't know how to use Well-Ordering Principle in Lean.

cairunze cairunze 4:54 PM

Maybe docs#iUnion_disjointed is something you want?
To use WOP, docs#Nat.find, docs#Nat.find_spec etc is useful.

👍 You

Arshak Parsa 6:18 PM

I somehow managed to prove this, thanks!

new members > Increasing Sequence of Sets induces Partition on Limit SEP 17

Arshak Parsa 1:56 PM

Can someone prove this?
(Is it even provable?)

```
import Mathlib

theorem iUnion_eq_iUnion_diff
  {A : ℕ → Set Ω} (hm : Monotone A) (h0 : A 0 = ∅) :
    (⋃ i, A i) = ⋃ i, A i \ A (i - 1) := by
  ext x
  constructor
  · intro h
    simp_all only [Set.mem_iUnion, Set.mem_diff]
    obtain ⟨w, h⟩ := h
    sorry
  · intro h
    simp_all only [Set.mem_iUnion, Set.mem_diff]
    obtain ⟨w, h⟩ := h
    obtain ⟨left, right⟩ := h
    apply Exists.intro
    · exact left
```

I found a proof here but I don't know how to use Well-Ordering Principle in Lean.

cairunze cairunze 4:54 PM

Maybe docs#iUnion_disjointed is something you want?
To use WOP, docs#Nat.find, docs#Nat.find_spec etc is useful.

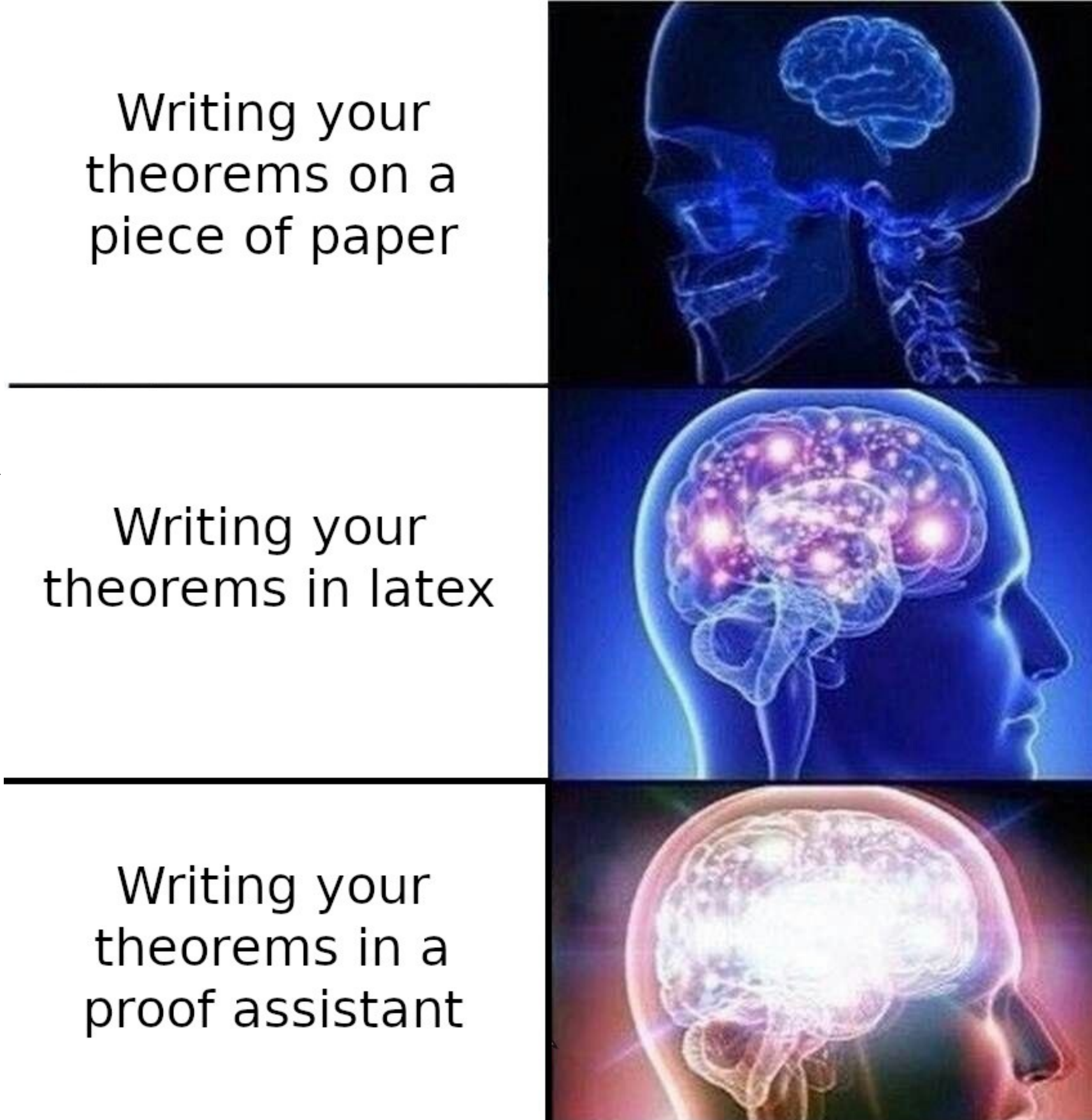
👍 You

Arshak Parsa 6:18 PM

I somehow managed to prove this, thanks!

What's the difference between Lean and Latex?

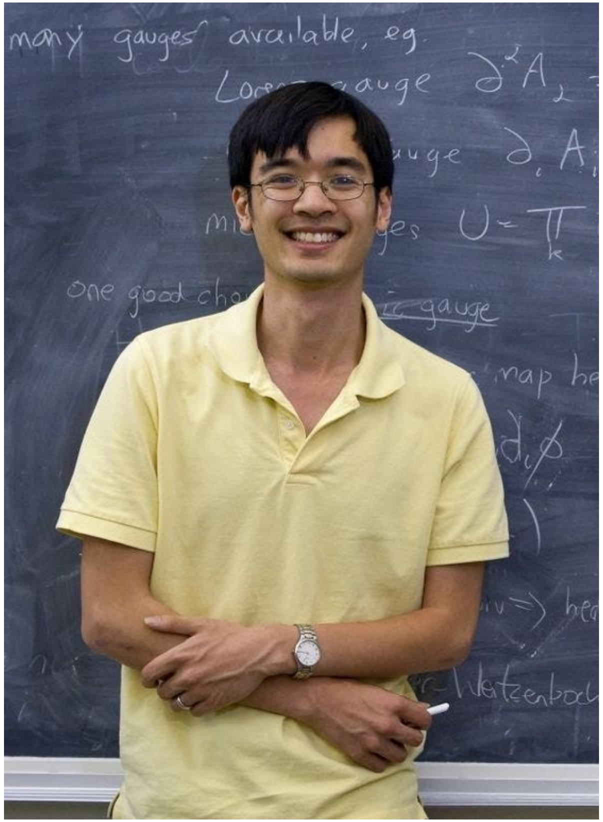
Latex can't check correctness of your proofs!



What can I do as a statistician?

- Use Lean as your proof assistant.
- Build AI models for Lean.
- Define statistical concepts in Lean.
- Prove CLT in Lean (CLT has been proven in Isabelle)

Does anyone use Lean?

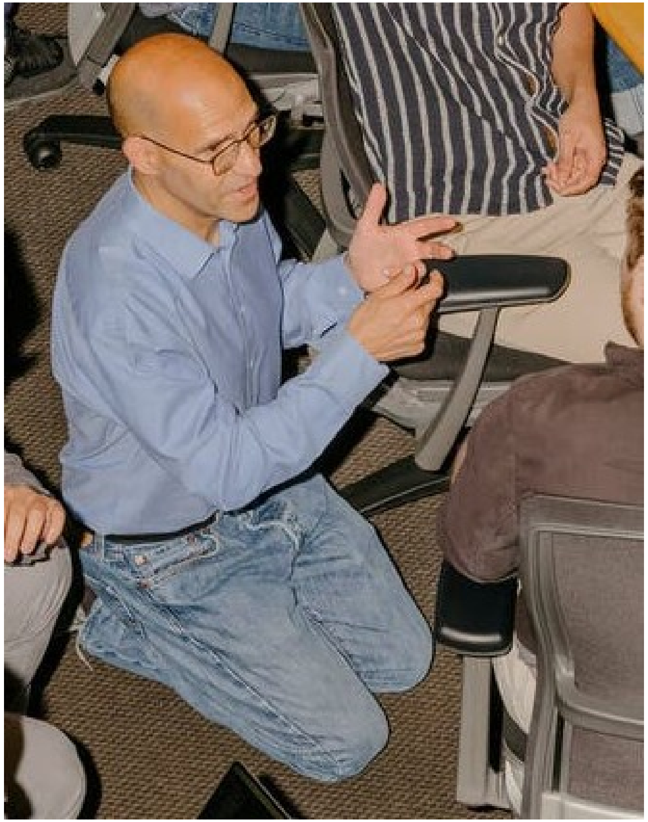


Terence Tao

Professor of Mathematics, UCLA

Kevin Buzzard

Professor of pure mathematics, Imperial College London



Jeremy Avigad

Professor of Philosophy and Mathematical Sciences, Carnegie Mellon University

And many more…

Lean is not alone!

- COQ
- Isabelle
- Agda
- And many more!

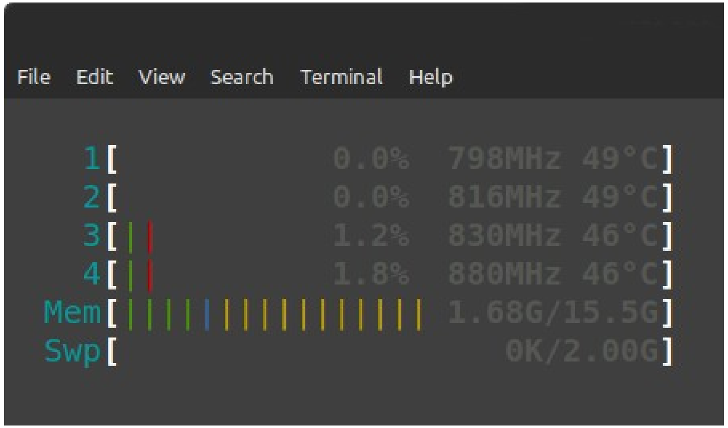
System requirements

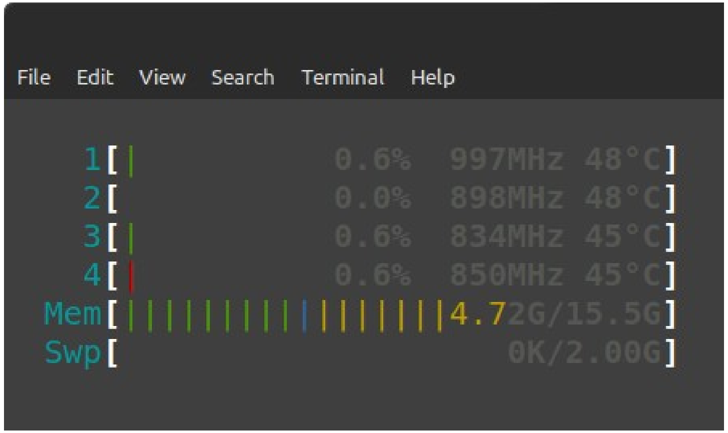
- Lean 4 only :

  2 GB RAM + 400 MB storage
- Lean 4 + Mathlib :

  6 GB RAM + 6 GB storage
- You can also use the web version:

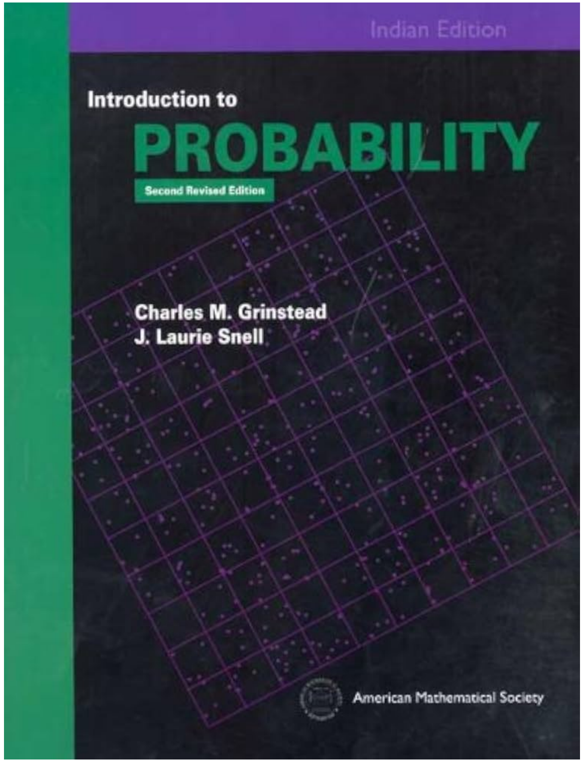
  https://live.lean-lang.org/





Learn more

- I am currently formalizing “Grinstead and Snell's Introduction to Probability” and it is on my github page: https://github.com/akp2003/prob-book
- You can ask lean questions on: https://proofassistants.stackexchange.com/ or on zulip chat: https://leanprover.zulipchat.com
- Watch “Lean for the Curious Mathematician 2022”



This book is open-source!