The Making of Lean

Leo de Moura
Senior Principal Applied Scientist, AWS
Chief Architect, Lean FRO

December 9, 2025

Before Lean, there was Z3

Z3 is a state-of-the-art SMT solver.

$a > 3, (a = b \vee a = b + 1), f(a) = 0, f(b) = 1$

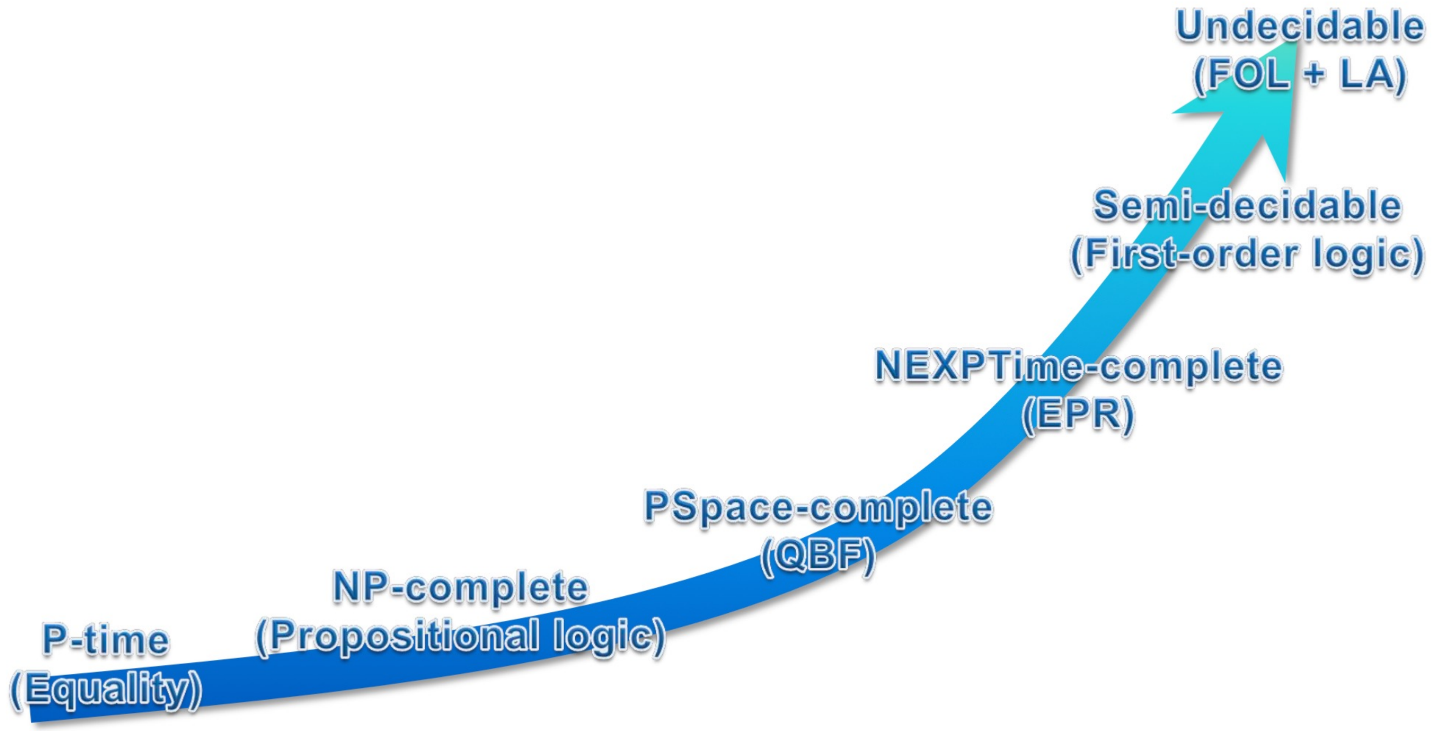
Model/Structure:

$a \rightarrow 4$

$b \rightarrow 3$

$f \rightarrow \{ 4 \rightarrow 0,\ 3 \rightarrow 1,\ \ldots \}$

In the SMT world, “Logic is calculus of computer science”



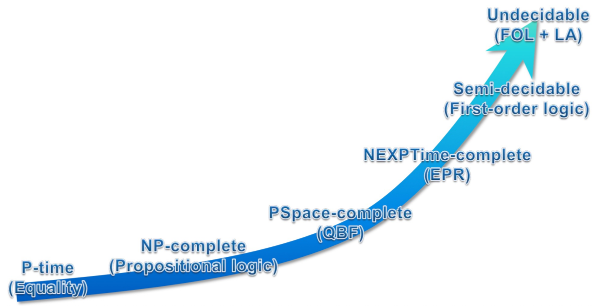
Before Lean, there was Z3

Z3 powered **bug-finding** pipelines like Microsoft's SAGE fuzzing tool caught thousands of defects.

Impact: since 2007

- **200+ machine years (in largest fuzzing lab in the world)**
- **1 Billion+ constraints (largest SMT solver usage ever!)**
- **100s of apps, 100s of bugs (missed by everything else…)**
- **Ex: 1/3 of all Win7 WEX security bugs found by SAGE**
- **Bug fixes shipped quietly (no MSRCs) to 1 Billion+ PCs**
- **Millions of dollars saved (for Microsoft and the world)**
- **SAGE is now used daily in Windows, Office, etc.**

Patrice Godefroid, SAGE's architect



Z3’s impact

Z3 is used by many research groups and big-tech companies.

Z3’s paper has more than 12,000 citations, and “most influential systems paper in the first 20 years of TACAS”

Ships with many other popular systems: Isabelle, SLAM/SDV, Visual Studio.

Received the Programming Languages Software Award from ACM SIGPLAN in 2015.

Note: I met Jeremy Avigad in 2007, when I taught a lecture in Ed Clark’s course about SAT/SMT. The technology behind Z3.

Z3's limitations

Z3 is also also used in **software verification** tools: Dafny, Verus, F*, VCC, ESC/Java 2, etc.

Users annotate programs with contracts, invariants, preconditions, etc.

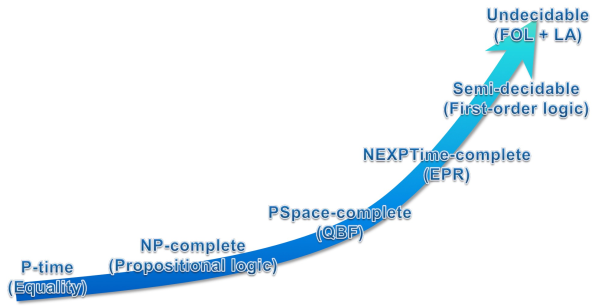
Program logic: Dijkstra's weakest precondition.

Verification condition generator converts annotations into Z3 queries.

Z3 input contains many universal quantifiers.

$$\begin{aligned}&\forall\, h,o,f:\\ &\quad \mathrm{IsHeap}(h) \land o \neq \mathrm{null} \land \mathrm{read}(h, o, \mathrm{alloc}) = t\\ &\quad \Rightarrow\\ &\quad \mathrm{read}(h,o, f) = \mathrm{null} \lor \mathrm{read}(h, \mathrm{read}(h,o,f),\mathrm{alloc}) = t\end{aligned}$$

$$\forall\, i,j:\ i \leq j \Rightarrow \mathrm{read}(a,i) \leq \mathrm{read}(b,j)$$



Z3’s limitations

Z3 is also also used in **software verification** tools: Dafny, Verus, F*, VCC, ESC/Java 2, etc.

Users annotate programs with contracts, invariants, preconditions, etc.

Program logic: Dijkstra’s weakest precondition.

Verification condition generator converts annotations into Z3 queries.

Main problems:

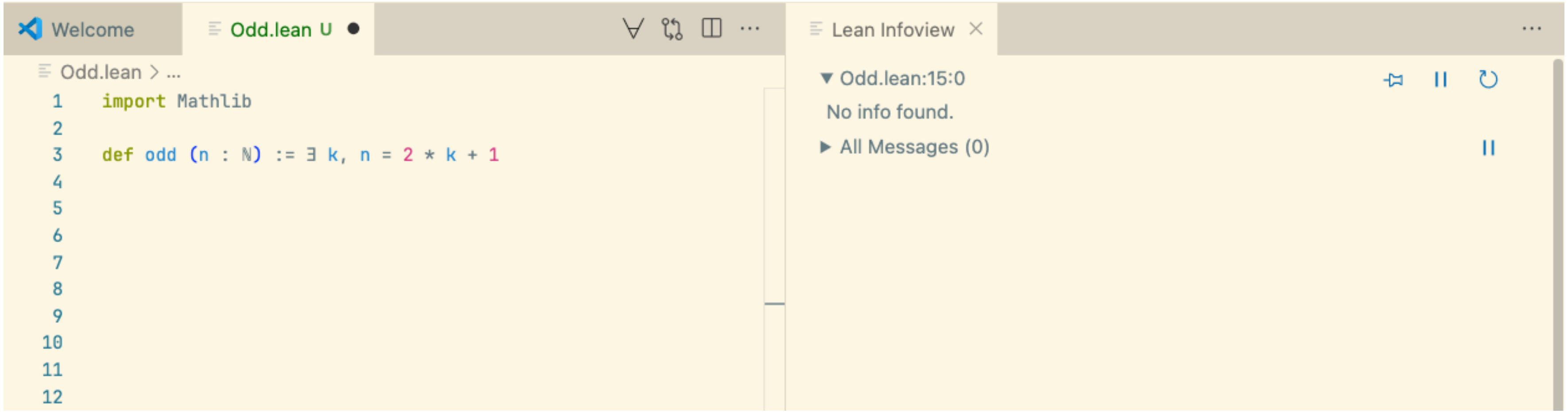
- **Proof instability**
- **Proof opaqueness**
- Scalability issues

Lean is an open-source programming language and proof assistant that is transforming how we approach mathematics, software verification, and AI.

The Lean project, started in 2013, aimed at merging interactive and automated theorem proving.

Lean provides **machine-checkable proofs**.

A small example



```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1
```

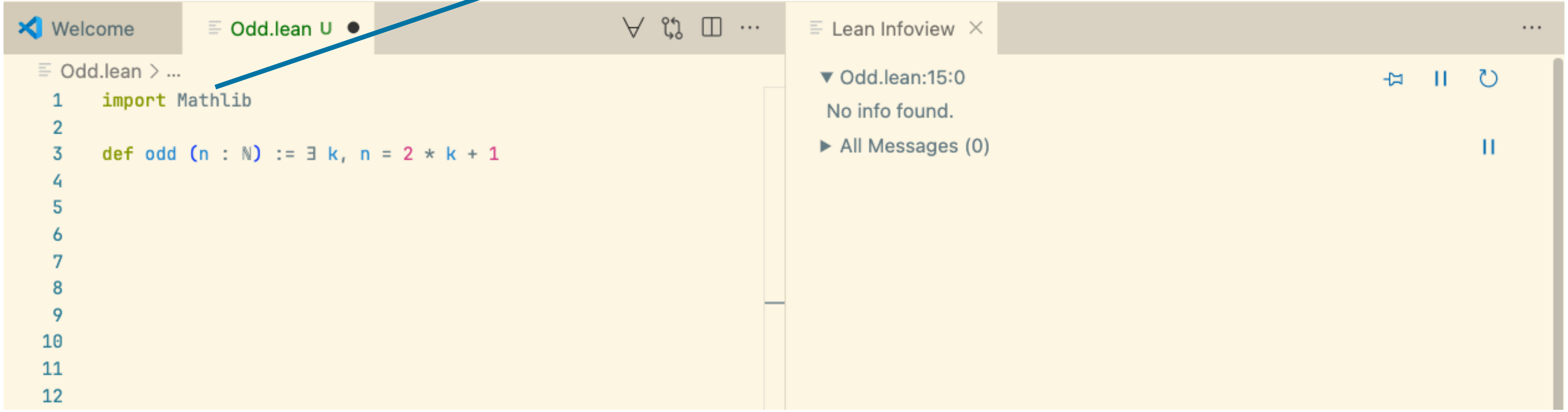
▼ Odd.lean:15:0

No info found.

▶ All Messages (0)

A small example

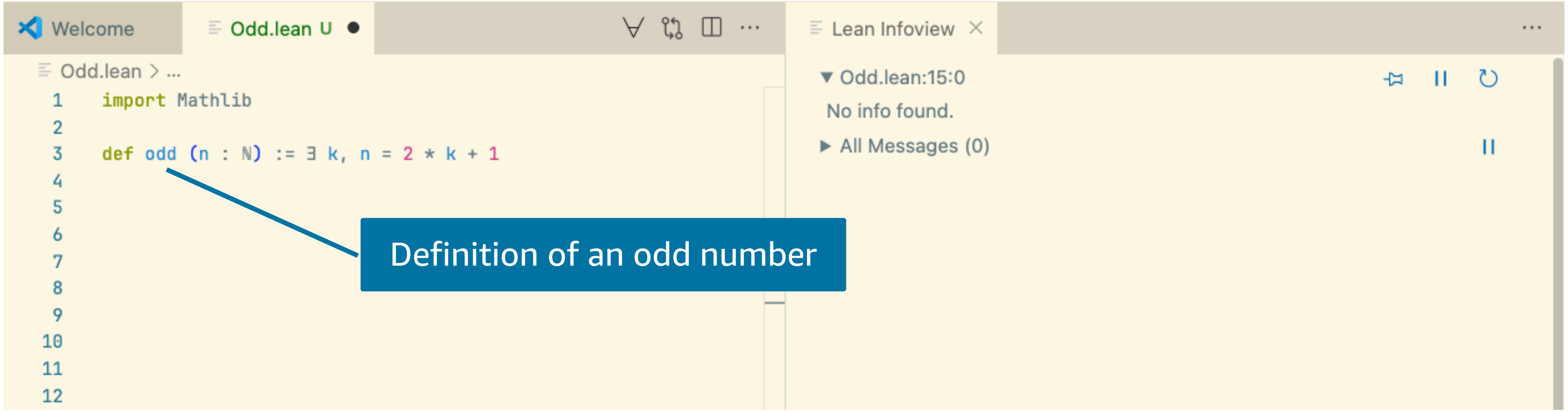
Mathlib is the Lean Mathematical library



```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1
```

```
▼ Odd.lean:15:0
No info found.
▶ All Messages (0)
```

A small example

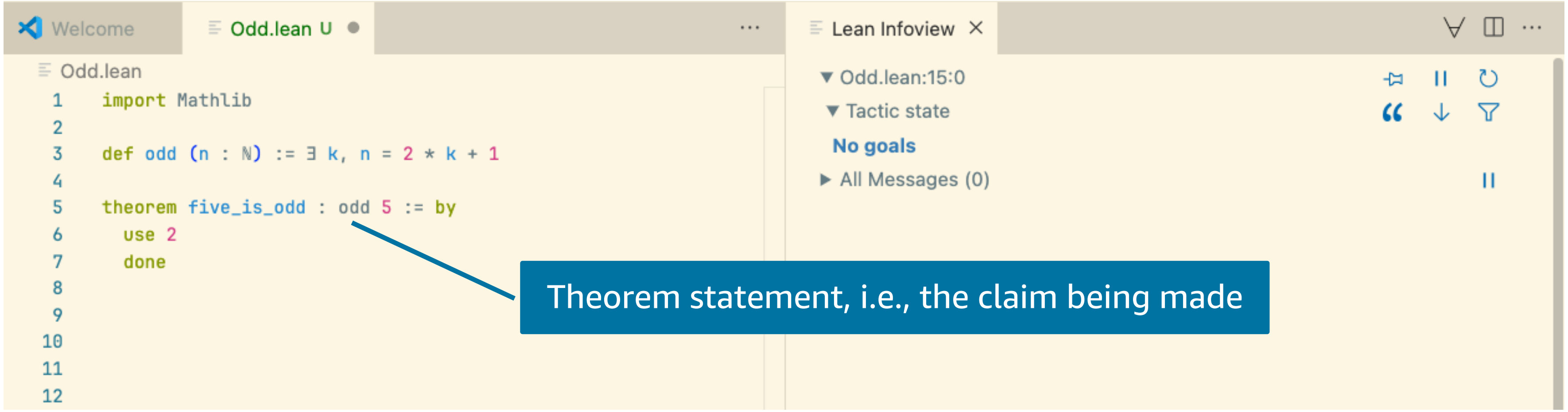


```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1
```

```
▼ Odd.lean:15:0
No info found.
▶ All Messages (0)
```

Definition of an odd number

Our first theorem

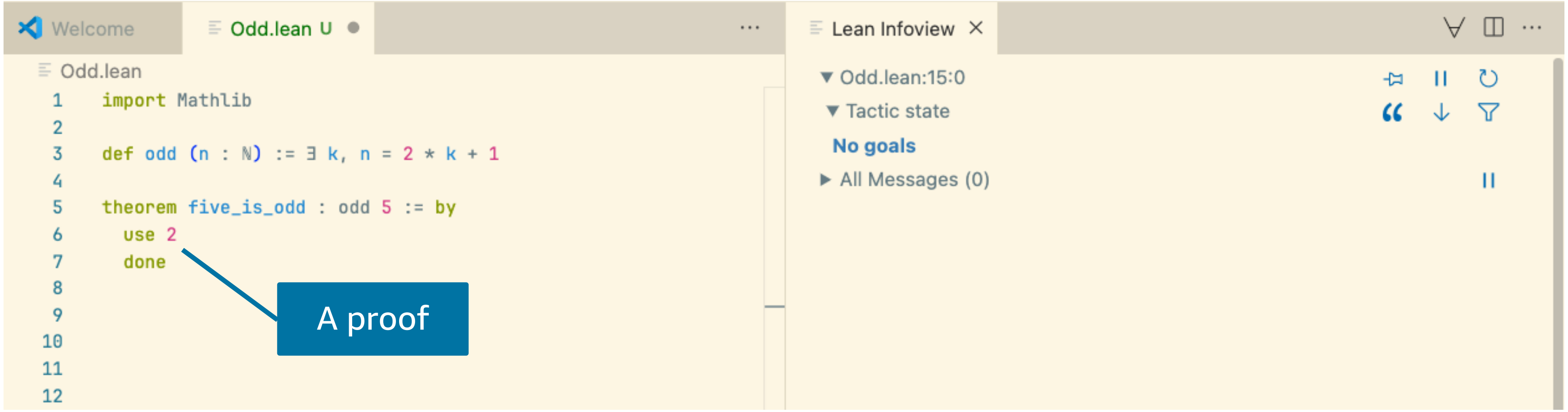


```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

theorem five_is_odd : odd 5 := by
  use 2
  done
```

```
▼ Odd.lean:15:0
 ▼ Tactic state
  No goals
▶ All Messages (0)
```

Theorem statement, i.e., the claim being made

Our first theorem

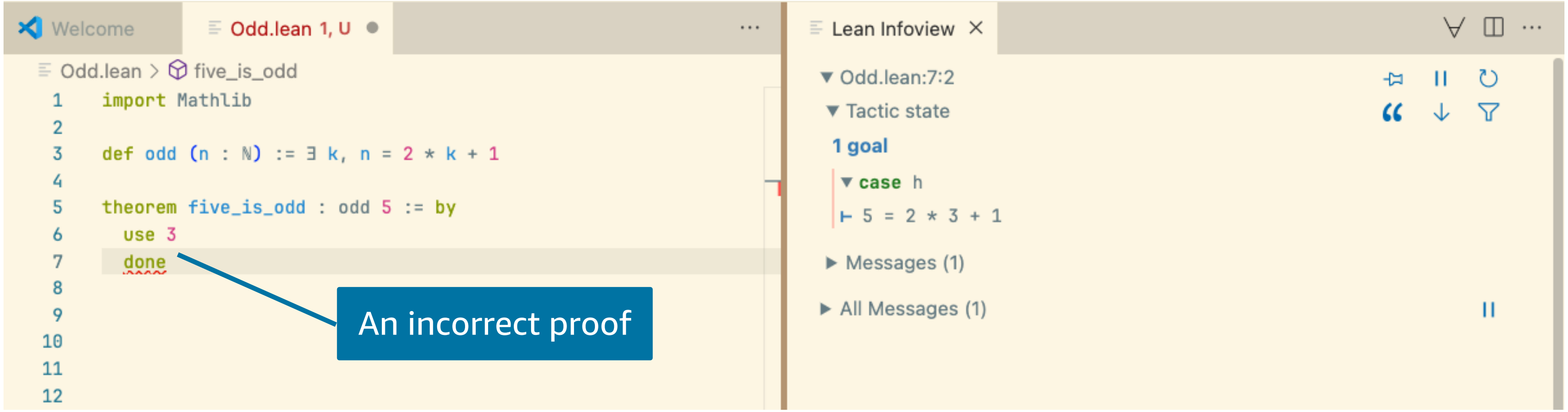


```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

theorem five_is_odd : odd 5 := by
  use 2
  done
```

A proof

```
▼ Odd.lean:15:0
  ▼ Tactic state
    No goals
▶ All Messages (0)
```

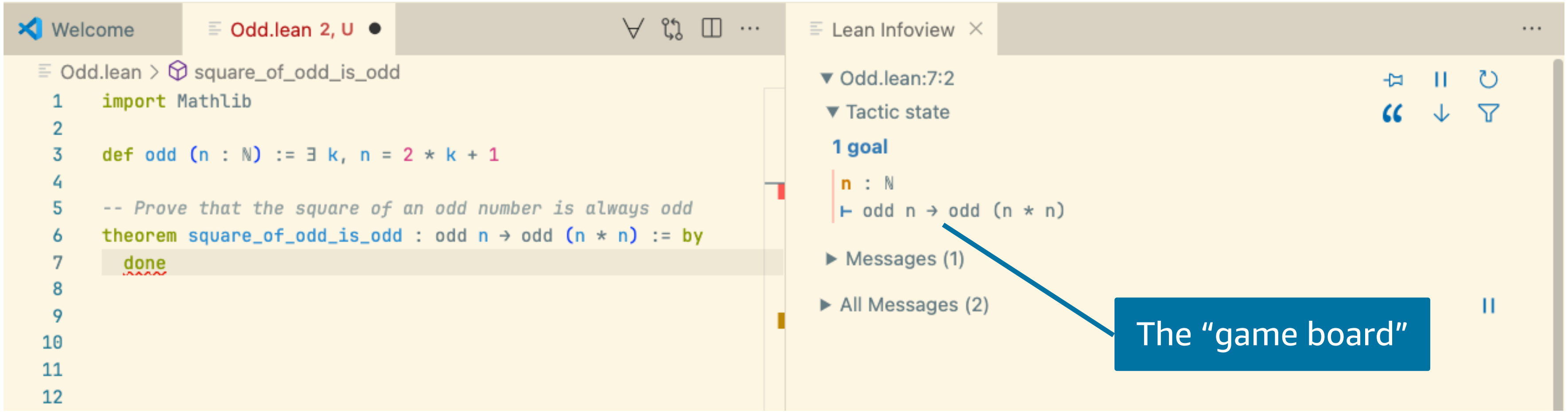
Our first theorem



```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

theorem five_is_odd : odd 5 := by
  use 3
  done
```

```
▼ Odd.lean:7:2
▼ Tactic state
1 goal
▼ case h
⊢ 5 = 2 * 3 + 1
▶ Messages (1)
▶ All Messages (1)
```

Theorem proving in Lean is an interactive game



Welcome | Odd.lean 2, U ●

Odd.lean > square_of_odd_is_odd

```lean
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

-- Prove that the square of an odd number is always odd
theorem square_of_odd_is_odd : odd n → odd (n * n) := by
  done
```

Lean Infoview

▼ Odd.lean:7:2

▼ Tactic state

1 goal

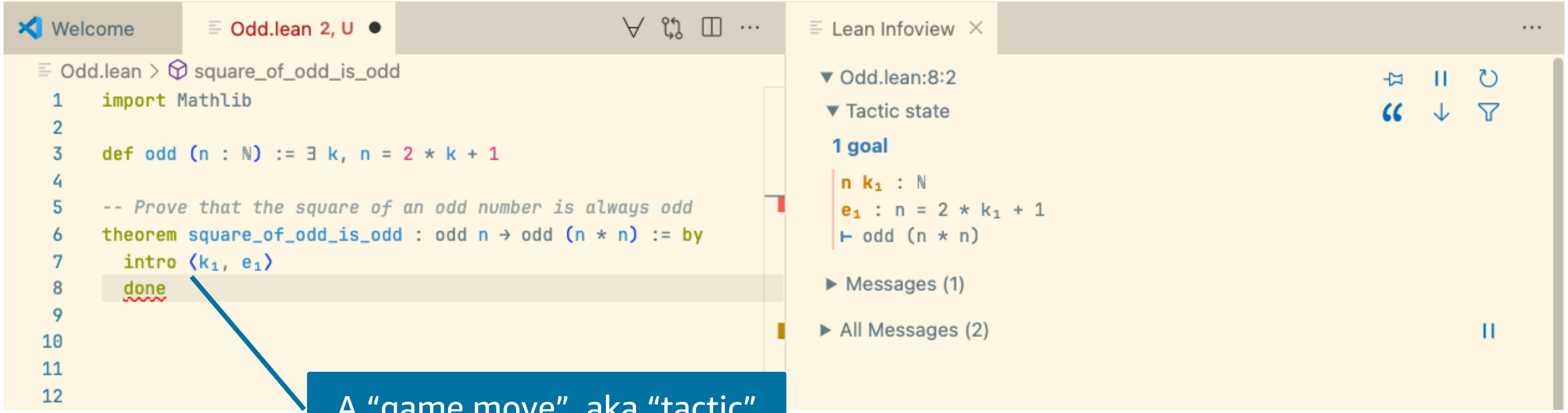
```
n : ℕ
⊢ odd n → odd (n * n)
```

▶ Messages (1)

▶ All Messages (2)

“You have written my favorite computer game”, Kevin Buzzard

Theorem proving in Lean is an interactive game

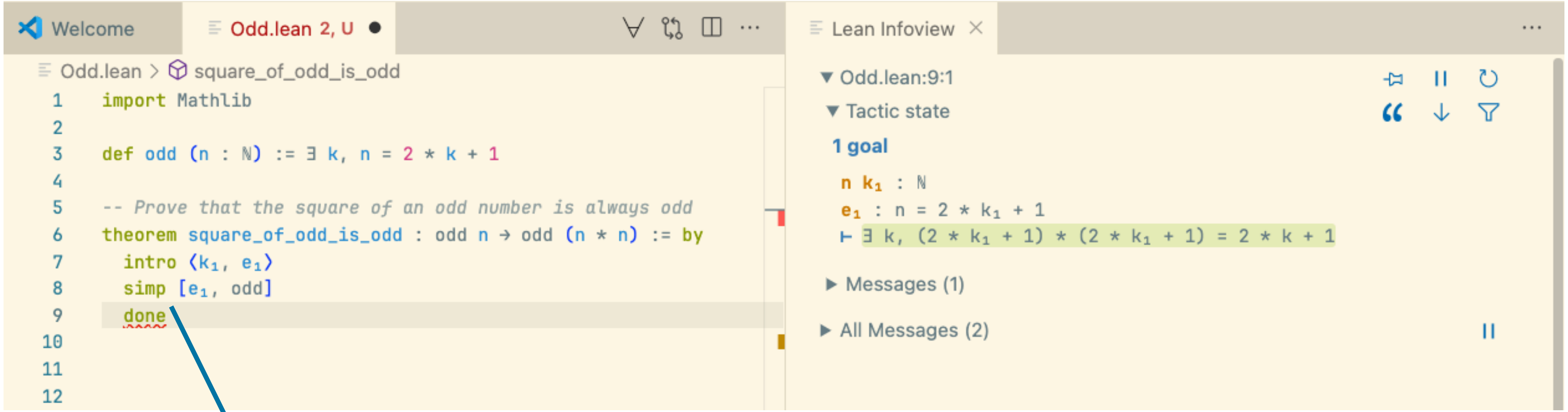


```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

-- Prove that the square of an odd number is always odd
theorem square_of_odd_is_odd : odd n → odd (n * n) := by
  intro ⟨k₁, e₁⟩
  done
```

```
▼ Odd.lean:8:2
 ▼ Tactic state
  1 goal
  n k₁ : ℕ
  e₁ : n = 2 * k₁ + 1
  ⊢ odd (n * n)
 ▶ Messages (1)
▶ All Messages (2)
```

A “game move”, aka “tactic”

Theorem proving in Lean is an interactive game

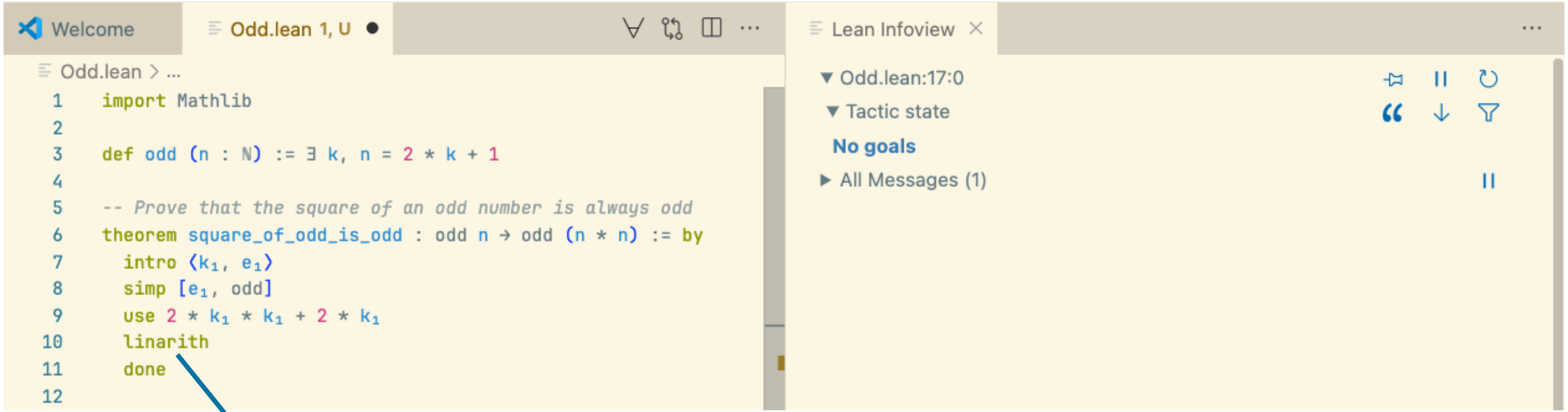


```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

-- Prove that the square of an odd number is always odd
theorem square_of_odd_is_odd : odd n → odd (n * n) := by
  intro ⟨k₁, e₁⟩
  simp [e₁, odd]
  done
```

```
▼ Odd.lean:9:1
▼ Tactic state
1 goal
n k₁ : ℕ
e₁ : n = 2 * k₁ + 1
⊢ ∃ k, (2 * k₁ + 1) * (2 * k₁ + 1) = 2 * k + 1
▶ Messages (1)
▶ All Messages (2)
```

The “game move” `simp`, the simplifier, is one of the most popular moves in our game

Theorem proving in Lean is an interactive game



Welcome | Odd.lean 1, U ● | Lean Infoview

Odd.lean > ...

```
import Mathlib

def odd (n : ℕ) := ∃ k, n = 2 * k + 1

-- Prove that the square of an odd number is always odd
theorem square_of_odd_is_odd : odd n → odd (n * n) := by
  intro ⟨k₁, e₁⟩
  simp [e₁, odd]
  use 2 * k₁ * k₁ + 2 * k₁
  linarith
  done
```

▼ Odd.lean:17:0
▼ Tactic state
No goals
▶ All Messages (1)

We complete this level using `linarith`, the linear arithmetic, move

Lean beginnings

Soonho Kong and I started coding in the **Fall of 2013**

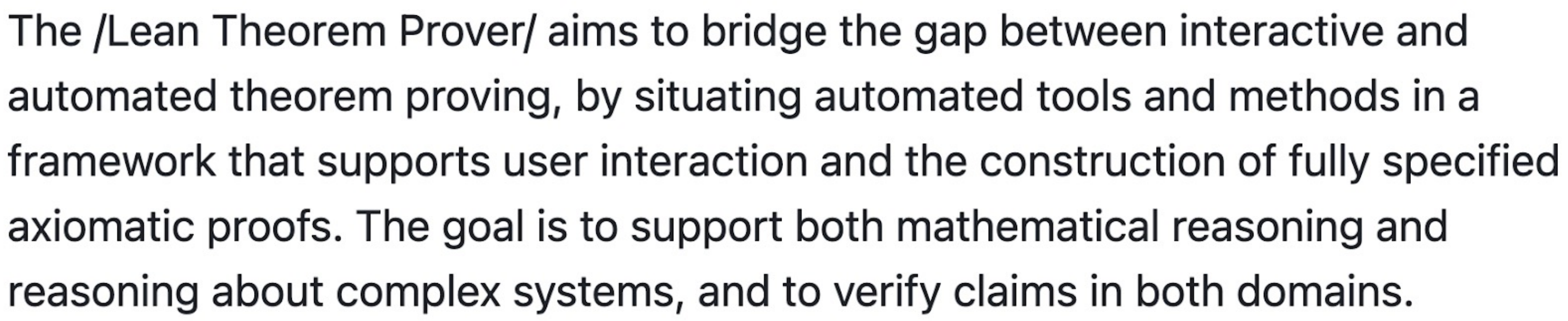
Platform for

- Software verification
- Formalized Mathematics
- Domain specific languages

First user after a few months: Jeremy Avigad (CMU)

Acknowledgement: Many thanks to Jeremy Avigad for digging up old messages between us.

Jeremy in July 2013: *Is there a story behind the name "Lean"?*
Me: *I want it to be smaller and simpler than Z3.*
Z3 is too big and has a lot of unnecessary optimizations.
I want the new code to be clean.



The /Lean Theorem Prover/ aims to bridge the gap between interactive and automated theorem proving, by situating automated tools and methods in a framework that supports user interaction and the construction of fully specified axiomatic proofs. The goal is to support both mathematical reasoning and reasoning about complex systems, and to verify claims in both domains.

From the “about” page in our original website at the beginning of 2014. Crafted by Jeremy, Soonho, and me

“When Leo had an idea, it was a good idea to let him pursue it” – Tom Ball, MSR Manager

Lean design philosophy

Issues should be closed as quickly as possible.

“Jeez, can’t they savor a new feature for a few hours before filing the next issue?” - Soonho

The kernel should be minimal. It should be easy to implement independent type checkers.

Lean should be extensible.

Support for dependent types.

“You can cry and scream and tear your hair out, but there is no way you can get the system [based on HOL] to admit that some beautiful theorem about the ring class applies to Zmod N.” - Jeremy Avigad

Failure is not an option.

Lean 0.1: no inductive datatypes

```
definition prim_rec_fun {A : (Type U)} (x : A) (f : A → num → A)
:= simp_rec (λ n : num, x) (λ fn n, f (fn (pre n)) n)

definition prim_rec {A : (Type U)} (x : A) (f : A → num → A) (m : num)
:= prim_rec_fun x f m (pre m)

theorem prim_rec_thm {A : (Type U)} (x : A) (f : A → num → A)
                    : prim_rec x f zero = x ∧
                      ∀ m, prim_rec x f (succ m) = f (prim_rec x f m) m
```

Lean 0.1 was “extensible” but using Lua

```
-- The following macro is used to create nary versions of operators such as mp.
-- Example: suppose we invoke
--
--   nary_macro("mp'", Const("mp"), 4)
--
-- Then, the macro expression
--
--    mp' Foo H1 H2 H3
--
-- produces the expression
--
--    (mp (mp (mp Foo H1) H2) H3)
function nary_macro(name, f, farity)
   local bin_app = function(e1, e2)
      local args = {}
      args[#args + 1] = f
      for i = 1, farity - 2, 1 do
         args[#args + 1] = mk_placeholder()
      end
      args[#args + 1] = e1
      args[#args + 1] = e2
      return mk_app(unpack(args))
   end
```

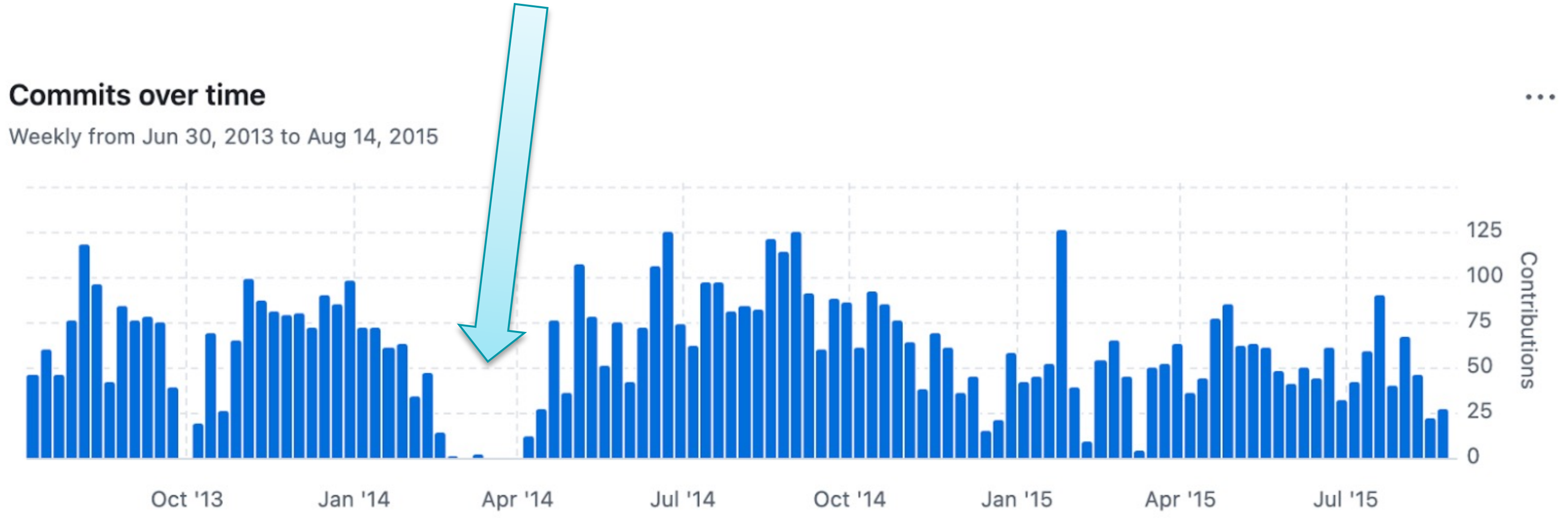
Lean 0.1 example from January 2014

```
theorem EvenPlusEven {a b : Nat} (H1 : even a) (H2 : even b) : even (a + b)
:= obtain (w1 : Nat) (Hw1 : a = 2*w1), from H1,
   obtain (w2 : Nat) (Hw2 : b = 2*w2), from H2,
     exists::intro (w1 + w2)
        (calc a + b  =  2*w1 + b      : { Hw1 }
                ...  =  2*w1 + 2*w2   : { Hw2 }
                ...  =  2*(w1 + w2)   : symm (distributer 2 w1 w2))
```

Lean 0.1 “It is a piece of crap”

Jeremy had already proved that there are infinite primes using 0.1 **but** constructing dependent pairs without inductive datatype support in the kernel was a disaster.

Build, rebuild … back to the drawing board



Lean 2

Elaboration as a constraint solving problem.

Soonho comes for another internship in the summer of 2014.

Inductive datatypes and recursive equations support.

The kernel was configurable, and we had support for HoTT.

Floris Van Doorn and Jakob von Raumer become major users.

Jeremy spends one month with us as a visiting researcher at Microsoft Research.

We started building a library for Lean.

Soonho built the Emacs lean-mode. *Soonho has the warrior spirit*.

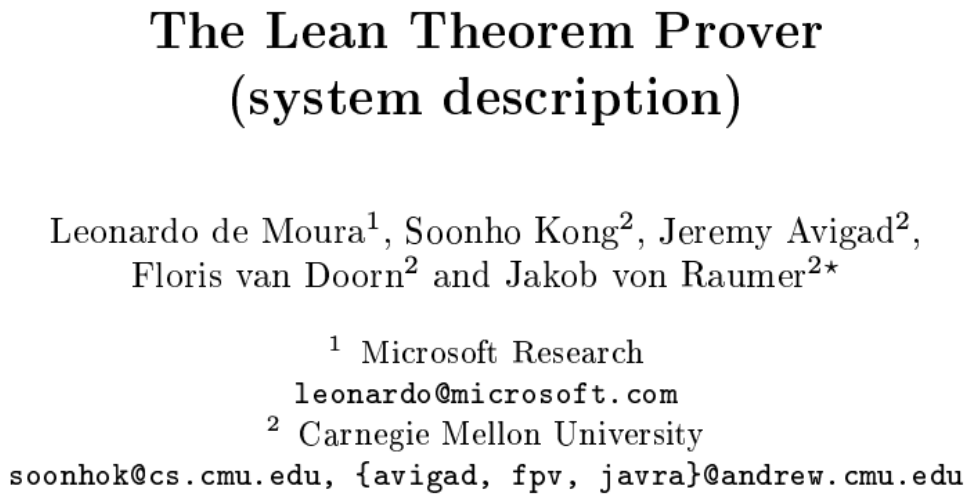
Ed Clarke (Soonho's PhD advisor): *"Let's teach a course using Lean"*

Lean 2

Our first paper about Lean is rejected. My first rejected paper in more than 10 years.

Reviewer: *"Why do we need another proof assistant?"*

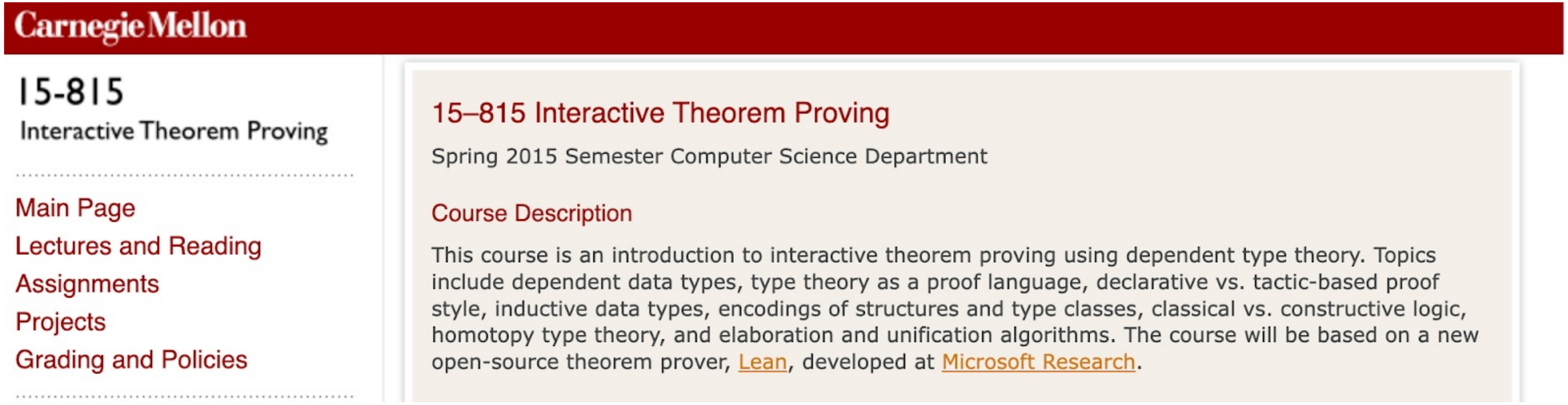
We succeed in submitting a system description to CADE 2015.



The Lean Theorem Prover (system description)

Leonardo de Moura[1], Soonho Kong[2], Jeremy Avigad[2], Floris van Doorn[2] and Jakob von Raumer[2*]

[1] Microsoft Research
leonardo@microsoft.com
[2] Carnegie Mellon University
soonhok@cs.cmu.edu, {avigad, fpv, javra}@andrew.cmu.edu



CarnegieMellon

15-815

Interactive Theorem Proving

Main Page

Lectures and Reading

Assignments

Projects

Grading and Policies

15–815 Interactive Theorem Proving

Spring 2015 Semester Computer Science Department

Course Description

This course is an introduction to interactive theorem proving using dependent type theory. Topics include dependent data types, type theory as a proof language, declarative vs. tactic-based proof style, inductive data types, encodings of structures and type classes, classical vs. constructive logic, homotopy type theory, and elaboration and unification algorithms. The course will be based on a new open-source theorem prover, Lean, developed at Microsoft Research.

Tom Hales came to every single lecture.

"Theorem Proving in Lean" is the lecture notes used in this course.

Jeremy, Soonho, and I created them on the fly.

First win: We got Tom excited about Lean.

Sebastian’s arrival

"Having done some toy formalization of my own in Lean in both algebraic and non-algebraic contexts by now, ***I am even more confident that this will be the first dependent type theory theorem prover I could imagine myself switching to from Isabelle.****"* – Sebastian to Jeremy on April 29, 2015

“Today I am crying of happiness” – My reaction to Sebastian’s first PR to the Lean code base on May 14, 2015

Meeting Daniel Selsam and Mario Carneiro at CICM

Message to Jeremy July 14, 2015

"***Daniel is an impressive guy***. I had lunch and dinner with him today. He has a degree in music and history, and only started to study Math and CS 6 years ago. ***He learned most of the material by himself*** using online course material and managed to get into MIT for masters and Stanford for PhD. ***He is super excited about Lean and has many projects in mind, most of them are based on machine learning*** …

BTW, ***I also met another young bright student, Mario Carneiro***. He is working on metamath … ***I think it would be great to "convert" him to Lean :-)***"

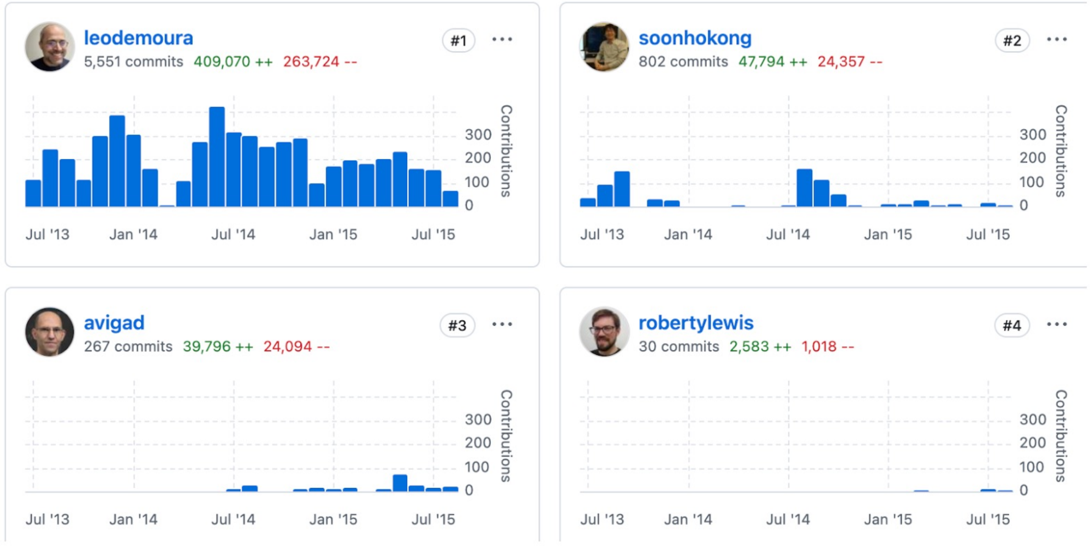
Retiring Lean 2

Adam Chlipala (MIT): “You need a real tactic framework in Lean”

Georges Gonthier and I submit a MSR expedition project: **Lean + AI for education**.

- Daniel and Jeremy contributed a lot to the project. We had a stellar team with many researchers from CMU and Stanford.
- **The project was not funded**.

After thousands of commits, we retire Lean 2



leodemoura #1
5,551 commits 409,070 ++ 263,724 --

soonhokong #2
802 commits 47,794 ++ 24,357 --

avigad #3
267 commits 39,796 ++ 24,094 --

robertylewis #4
30 commits 2,583 ++ 1,018 --

A Lean 2 example

```
theorem first_sylow_theorem {p : nat} (Pp : prime p) :
  ∀ n, p^n ∣ card G → ∃ (H : finset G) (finsubgH : is_finsubg H), card H = p^n
| 0        := assume Pdvd, exists.intro (singleton 1)
  (exists.intro one_is_finsubg
    (by rewrite [card_singleton, pow_zero]))
| (succ n) := assume Pdvd,
  obtain H PfinsubgH PcardH, from first_sylow_theorem n (pow_dvd_of_pow_succ_dvd Pdvd),
  assert Ppsubg : psubg H p n, from and.intro Pp PcardH,
  assert Ppowsucc : p^(succ n) ∣ (card (lcoset_type univ H) * p^n),
    by rewrite [-PcardH, -(lagrange_theorem' !subset_univ)]; exact Pdvd,
  assert Ppdvd : p ∣ card (lcoset_type (normalizer H) H), from
    dvd_of_mod_eq_zero
      (by rewrite [-(card_psubg_cosets_mod_eq Ppsubg), -dvd_iff_mod_eq_zero];
      exact dvd_of_pow_succ_dvd_mul_pow (pos_of_prime Pp) Ppowsucc),
  obtain J PJ, from cauchy_theorem Pp Ppdvd,
  exists.intro (fin_coset_Union (cyc J))
    (exists.intro _
      (by rewrite [pow_succ, -PcardH, -PJ]; apply card_Union_lcosets))
```

Lean 3 development starts in 2016

Simpler elaborator without backtracking search.

Daniel Selsam comes for an internship in the summer of 2016.

I decide to **use Lean itself to write a tactic framework for Lean**.

Daniel and I also started building “blast” a tactic based on the SMT solving techniques used in Z3.

Jeremy comes back as a visiting researcher for 4 weeks.

Gabriel Ebner joins the team.

Jeremy to Gabriel on Aug 7, 2016: *“Work is proceeding nicely on Lean 3, a major rewrite of the current Lean. It is *really* cool. One big new feature is that there is a fast evaluator, so that you can use Lean as a programming language. It is faster than interpreted Ocaml and Python!”*

Gabriel: *“This is pretty nice. Is there any description that I can read up on? I have quickly looked over the code, and it seems to be a stack machine that implements a call-by-value evaluation order?”*

Lean 3 – HoTT support was removed

Sometimes you disappoint your users.

We were basically implementing two systems.

We did not have resources. I was the only person working full-time on Lean.

Lean was my **research project** at Microsoft Research. **It was not a Microsoft Product.**

Floris maintained Lean 2 with HoTT support until 2018.

Gabriel wrote a hack to simulate HoTT in Lean 3.

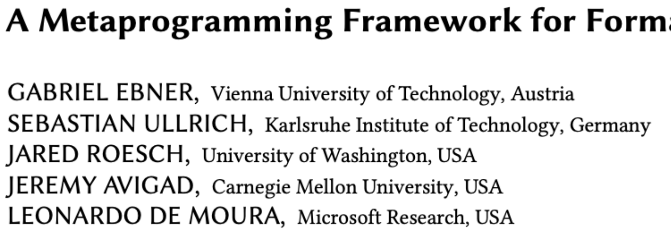
Lean 3 – POPL 2017 in Paris

First public demonstration of Lean 3.

Gabriel Ebner, Jared Roesch, Sebatian Ullrich, and I presented a Lean tutorial at POPL.

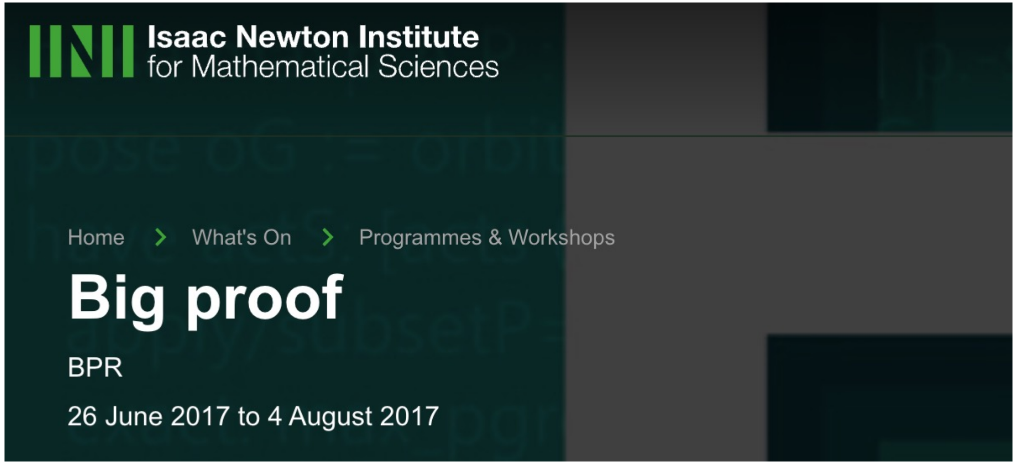
Jared was a PhD student at UW.

We also wrote a paper about the **new metaprogramming framework** for ICFP, and it becomes popular very quickly. We don't use Lua anymore for extensibility.



A Metaprogramming Framework for Formal Verification

GABRIEL EBNER, Vienna University of Technology, Austria
SEBASTIAN ULLRICH, Karlsruhe Institute of Technology, Germany
JARED ROESCH, University of Washington, USA
JEREMY AVIGAD, Carnegie Mellon University, USA
LEONARDO DE MOURA, Microsoft Research, USA


Isaac Newton Institute for Mathematical Sciences

Home > What's On > Programmes & Workshops

Big proof

BPR

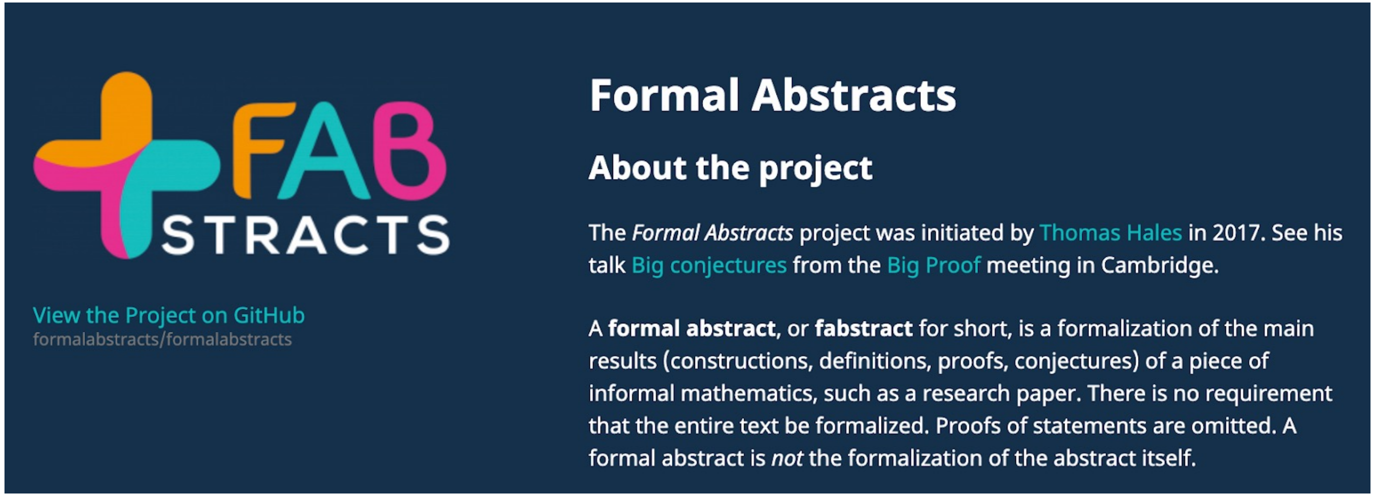
26 June 2017 to 4 August 2017

Organisers

- Jeremy Avigad *Carnegie Mellon University*
- Leonardo de Moura *Microsoft (USA)*
- Georges Gonthier *INRIA Saclay - Île-de-France*
- Ursula Martin *University of Oxford*
- J Strother Moore *University of Texas at Austin; University of Edinburgh*
- Lawrence Paulson *University of Cambridge*
- Andrew Pitts *University of Cambridge*
- Natarajan Shankar *SRI International*

I presented Lean 3 at Big Proof; it was well received.

But more importantly, Tom Hales presented


View the Project on GitHub
formalabstracts/formalabstracts

Formal Abstracts

About the project

The *Formal Abstracts* project was initiated by Thomas Hales in 2017. See his talk Big conjectures from the Big Proof meeting in Cambridge.

A **formal abstract**, or **fabstract** for short, is a formalization of the main results (constructions, definitions, proofs, conjectures) of a piece of informal mathematics, such as a research paper. There is no requirement that the entire text be formalized. Proofs of statements are omitted. A formal abstract is *not* the formalization of the abstract itself.

Tom announces he will do it in Lean. After his talk, several people surrounded Tom trying to convince him to use other systems.

Kevin Buzzard watched Tom's talk remotely.

Tensions with the community are rising…

Jeremy: *"How are things going with you?"*

Me: *"I'm surviving here. I have been trying to address issues raised by the KreMLin project… **I'm not sure they will be able to complete the compiler and verify it using Lean, but at least it is not a total disaster.** Armael is not super disappointed with Lean anymore. He found so many issues :(…*
***I need success stories to tell upper management**. So, it would be great to have Tom using Lean.*
…
***I'm ignoring Kim. It is sad, but it is the only way I found to recover my sanity.** Otherwise, I have to spend 100% of my time with them."*

Mathlib moves to its own repository

We made the decision at the end of Big Proof.

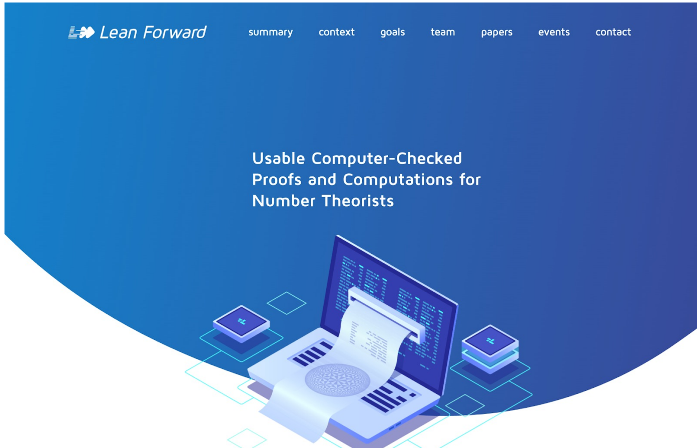
Development model: Cathedral (Lean) vs Bazar (Mathlib).

Many heated exchanges with Mario.

There were many threats of forking the Lean code base.

Fought with Jasmin Blanchette about unreasonable expectations.

Me: *"From my point of view, they are a bunch of entitled brats."*


Lean Forward
summary
context
goals
team
papers
events
contact
Usable Computer-Checked Proofs and Computations for Number Theorists

David Christiansen visits MSR in 2017

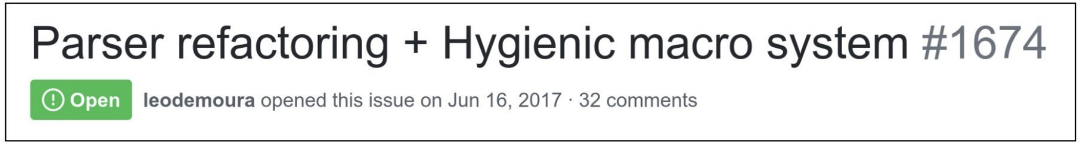
David gives a copy of his book: The Little Typer.

He implements the precursor of code actions in the lean-mode for Emacs.

It becomes clear that we need a way to represent Syntax in Lean.

In Lean 3, we simulate them using the kernel expressions themselves.

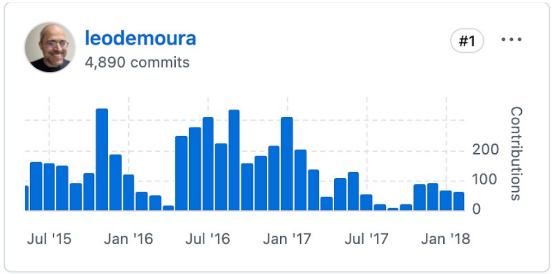
Limitation: We cannot reliably implement a code action that generates a match-expression.

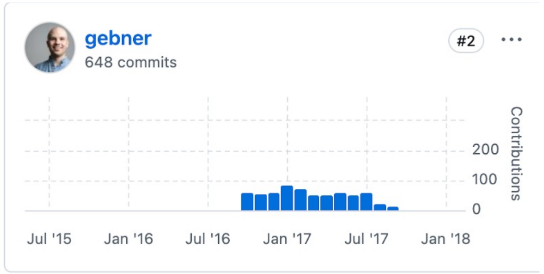

Parser refactoring + Hygienic macro system #1674

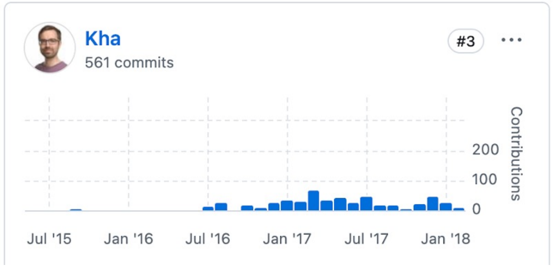
Open leodemoura opened this issue on Jun 16, 2017 · 32 comments

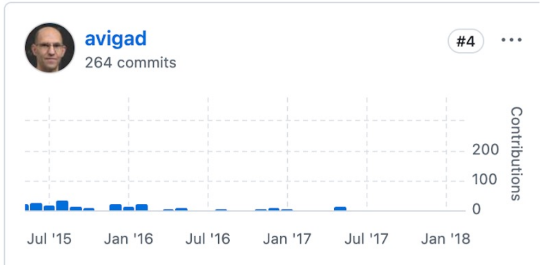
Retiring Lean 3…

After thousands additional commits, we thought we were retiring Lean 3.


leodemoura
4,890 commits
#1


gebner
648 commits
#2


Kha
561 commits
#3

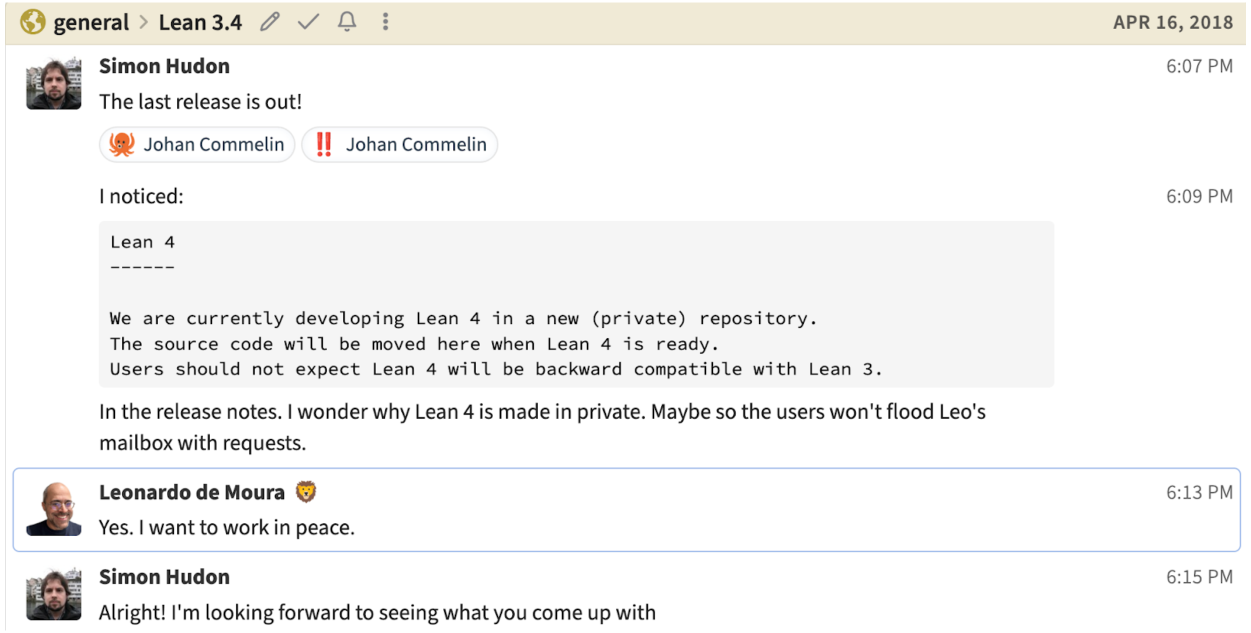

avigad
264 commits
#4

Lean 4 plans start at the beginning of 2018

Following the metaprogramming popularity, I decide to **implement Lean in Lean**.

Sebastian is ready for the big adventure.

There are many tensions with the community, and I decide to make the **Lean 4 repo private**.



general > Lean 3.4 APR 16, 2018

Simon Hudon 6:07 PM

The last release is out!

🐙 Johan Commelin ‼️ Johan Commelin

I noticed: 6:09 PM

```
Lean 4
------

We are currently developing Lean 4 in a new (private) repository.
The source code will be moved here when Lean 4 is ready.
Users should not expect Lean 4 will be backward compatible with Lean 3.
```

In the release notes. I wonder why Lean 4 is made in private. Maybe so the users won't flood Leo's mailbox with requests.

Leonardo de Moura 🦁 6:13 PM

Yes. I want to work in peace.

Simon Hudon 6:15 PM

Alright! I'm looking forward to seeing what you come up with

Lean 4: Stealth mode

It was amazing to work without distractions.

Many ideas for the code generator, Syntax, macro system, elaboration, etc.

We are making **Lean 4 very extensible.**

I am still super bitter with the community, and don't go to the first two Lean Together events.

Sebastian gives an update on Lean 4 in these two events.

Gabriel heroically maintains Lean 3 while we develop Lean 4. The Lean 3 community version is born.

Another Crisis

In 2019, Tom Ball became an IC again. He was my manager and the one giving me cover.

Daniel Selsam saved the project. He proposed the IMO Grand Challenge and announced he is going to use Lean. The MSR director loved the idea. Daniel joined MSR.

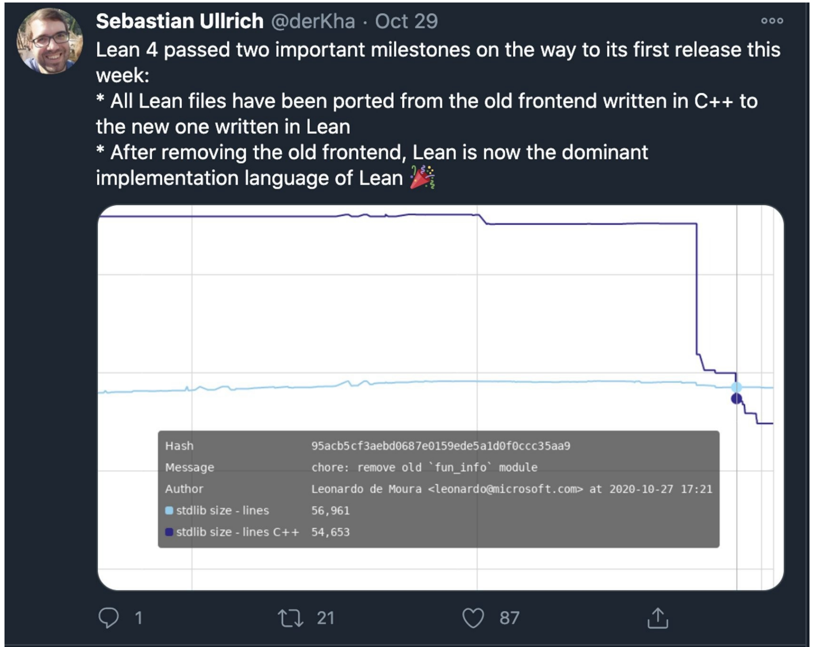
IMO Grand Challenge

Later that year, **Kevin visited MSR and gave a spectacular talk about formal mathematics** and Lean's relevance. It still is one of the most watched MSR talks ever. Daniel, Kevin, and I discussed the IMO GC.

Eureka, we compiled Lean 4 using Lean 4

In October 2020, we compile Lean 4 using Lean 4.

The code generator is still in C/C++, but the parser, elaborator, macro system, and tactic framework are all written in Lean itself.

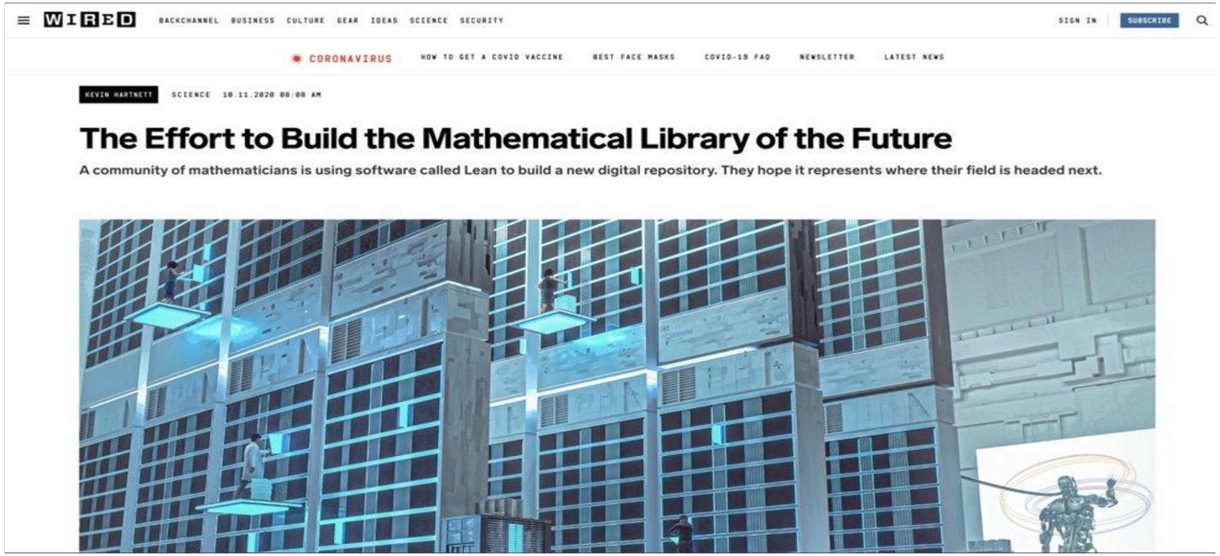


Sebastian Ullrich @derKha · Oct 29

Lean 4 passed two important milestones on the way to its first release this week:
* All Lean files have been ported from the old frontend written in C++ to the new one written in Lean
* After removing the old frontend, Lean is now the dominant implementation language of Lean 🎉

Hash	95acb5cf3aebd0687e0159ede5a1d0f0ccc35aa9
Message	chore: remove old `fun_info` module
Author	Leonardo de Moura <leonardo@microsoft.com> at 2020-10-27 17:21
🟦 stdlib size - lines	56,961
🟦 stdlib size - lines C++	54,653

In the meantime, Mathlib goes viral



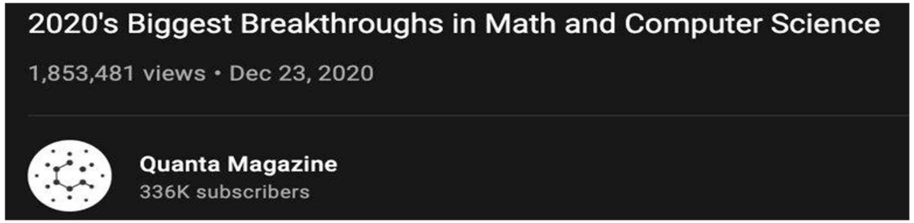
WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY SIGN IN SUBSCRIBE

CORONAVIRUS HOW TO GET A COVID VACCINE BEST FACE MASKS COVID-19 FAQ NEWSLETTER LATEST NEWS

KEVIN HARTNETT SCIENCE 10.11.2020 08:00 AM

The Effort to Build the Mathematical Library of the Future

A community of mathematicians is using software called Lean to build a new digital repository. They hope it represents where their field is headed next.



2020's Biggest Breakthroughs in Math and Computer Science

1,853,481 views • Dec 23, 2020

Quanta Magazine

336K subscribers

The Liquid Tensor Experiment

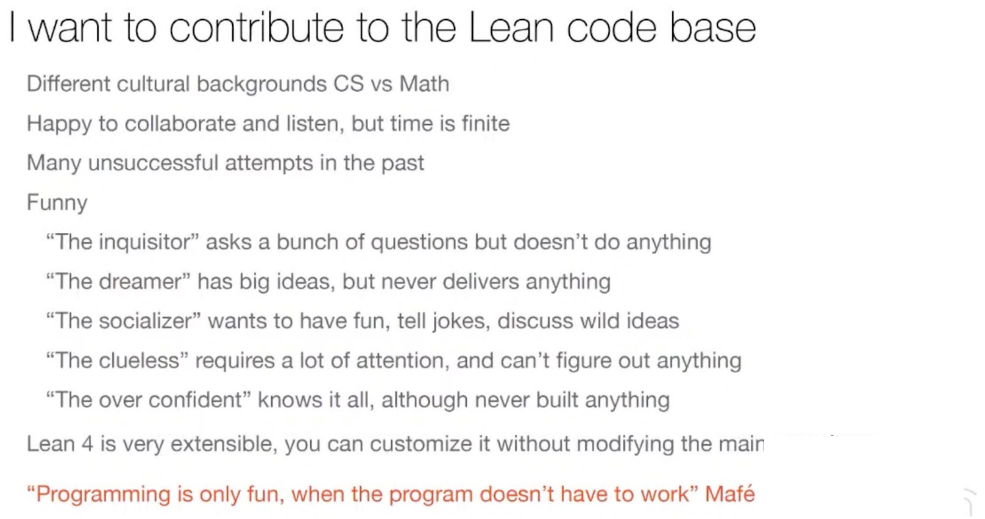
Nov 2020: Peter Scholze posits formalization challenge

“I spent much of 2019 obsessed with the proof of this theorem, almost getting crazy over it. In the end, we were able to get an argument pinned down on paper, but I think nobody else has dared to look at the details of this, and so I still have some small lingering doubts.”

Lean 4 is publicly announced at Lean Together 2021

Marc Huisinga hacks a VS Code extension for Lean 4 in record time.

I am still bitter with the community, but I attend Lean Together



I want to contribute to the Lean code base

Different cultural backgrounds CS vs Math

Happy to collaborate and listen, but time is finite

Many unsuccessful attempts in the past

Funny

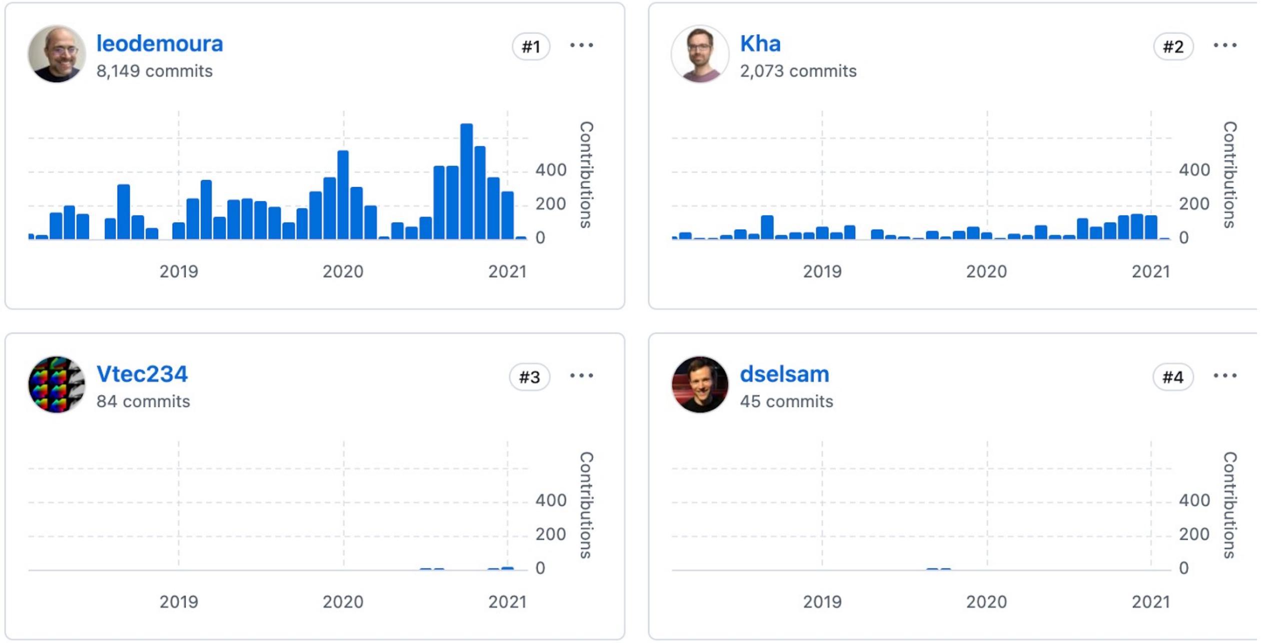
- "The inquisitor" asks a bunch of questions but doesn't do anything
- "The dreamer" has big ideas, but never delivers anything
- "The socializer" wants to have fun, tell jokes, discuss wild ideas
- "The clueless" requires a lot of attention, and can't figure out anything
- "The over confident" knows it all, although never built anything

Lean 4 is very extensible, you can customize it without modifying the mair

"Programming is only fun, when the program doesn't have to work" Mafé

During a session with Kevin and others, I urge the community that we need success stories to keep the project alive.

It is the end of the stealth mode era…


leodemoura
8,149 commits
#1

Kha
2,073 commits
#2

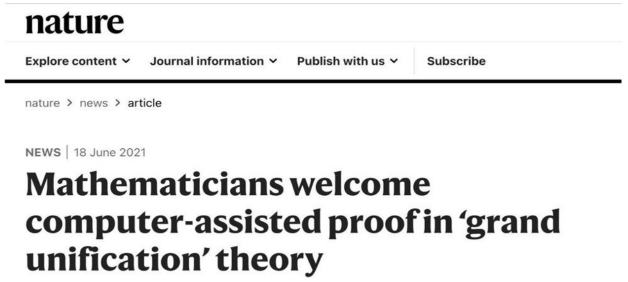
Vtec234
84 commits
#3

dselsam
45 commits
#4

The First Major Victory a few months later

Johan Commelin led a team with several members of the **Lean community and announced the formalization of the crucial intermediate lemma** that Scholze was unsure about, with only minor corrections, in **May 2021**.

> *"[T]his was precisely the kind of oversight I was worried about when I asked for the formal verification. [...] The proof walks a fine line, so if some argument needs constants that are quite a bit different from what I claimed, it might have collapsed"*, Peter Scholze


nature

Explore content Journal information Publish with us Subscribe

nature > news > article

NEWS | 18 June 2021

Mathematicians welcome computer-assisted proof in 'grand unification' theory

MathPort

Daniel needs Mathlib for the IMO GC, and starts developing MathPort

Mario comes for an internship with Daniel to work on MathPort.

Meeting Sam Altman

Daniel Selsam and I meet Sam Altman at the end of 2021.

Sam starts the meeting saying: “I love Lean …”

OpenAI is trying to solve the IMO GC using Lean.

Daniel leaves MSR a few weeks later and joins OpenAI.

Beginning of 2022 is great for Lean

MSR leadership is excited about Lean’s success.

Meta is also trying to use Lean to solve the IMO GC.

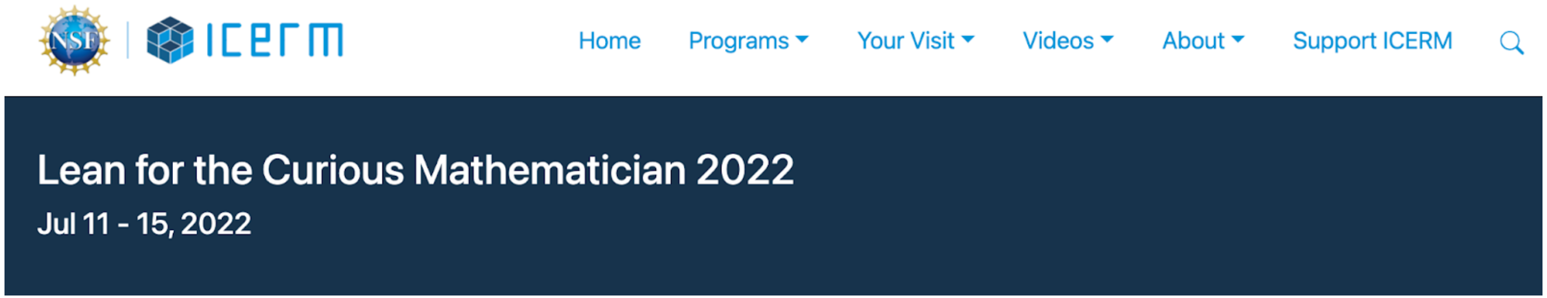
I get headcounts for the project: 1 PM and 2 RSDEs (Gabriel and Sebastian).

Mac comes for an internship to work on Lake.

I also have a budget to hire David Christiansen to write: “Functional Programming in Lean.”

MSR also provides small research grants to several Lean community members.

Finally made peace with the community



Lean for the Curious Mathematician 2022

Jul 11 - 15, 2022

A few weeks after the event, I hired Mario and Kim to port the Mathlib tactics to Lean 4.

MathPort cannot help with this part.

Achieving the Unthinkable

The full challenge was completed in July 2022.

The team not only verified the proof but also simplified it.
Moreover, they did this without fully understanding the entire proof.

Johan, the project lead, reported that he could only see two steps ahead. **Lean was a guide**.

> *"The Lean Proof Assistant was really that: an assistant in navigating through the thick jungle that this proof is. Really, one key problem I had when I was trying to find this proof was that I was essentially unable to keep all the objects in my RAM, and I think the same problem occurs when trying to read the proof"*, Peter Scholze

Another crisis…

August 2022, OpenAI leadership demos GPT4 to the Microsoft leadership.

In a few months, everything is about AI.

There is also an economic downturn.

Gabriel signed the contract before the crisis, but the headcount for Sebastian is cancelled.

Gabriel arrives in the middle of the storm.

There are also many doubts whether Mathlib will be ported to Lean 4 or not.

Crisis continues at the beginning of 2023

Mike Freedman sends a message to the leadership saying that Lean is one of the most important projects Microsoft Research has ever produced. The message is ignored.

Sebastian and I are working on the Lean FRO proposal, but it is unclear whether it will happen or not.

It is unclear whether we will have funding to continue maintaining the project. Jeremy tries to cheer me up: **"It was a good run, we learned a lot".**

Mike Freedman and I leave MSR on the same day in March 2023.

I join AWS a few weeks later.

Soonho is back

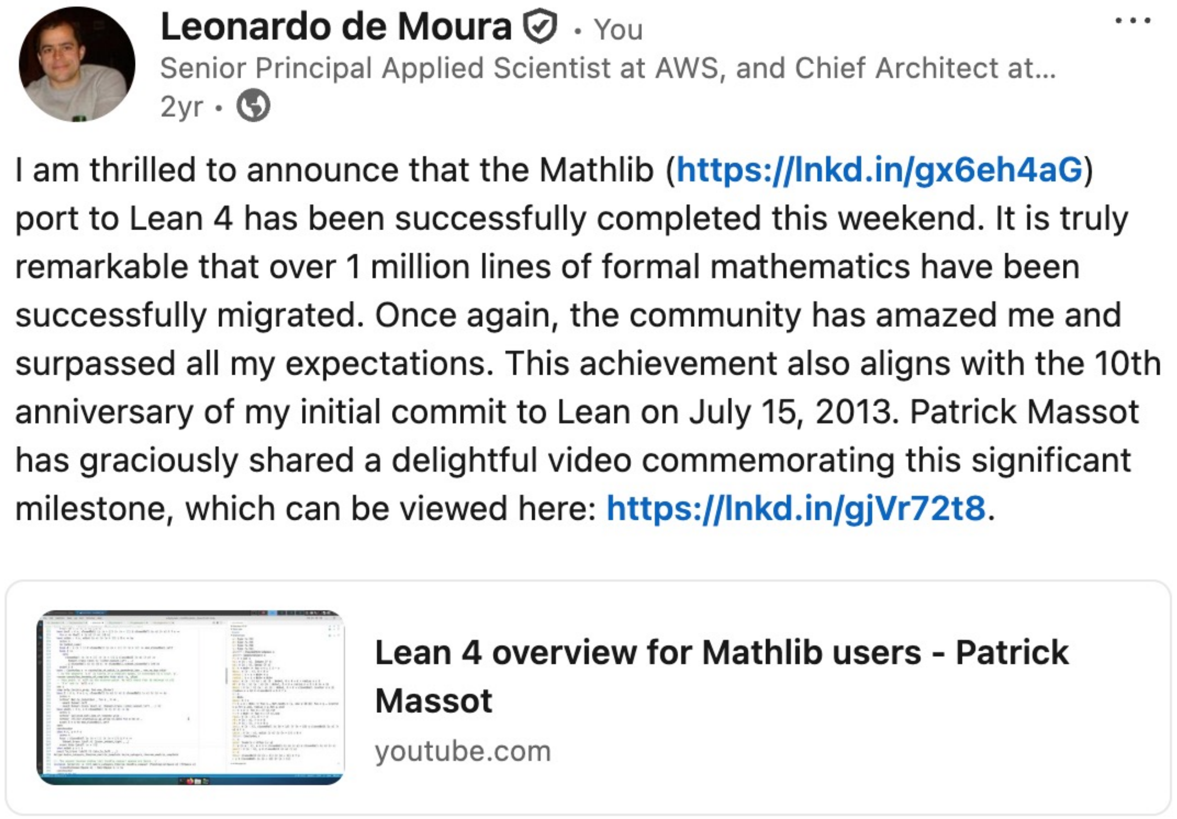
Soonho also works at AWS. We start a massive campaign to promote Lean at AWS.

Emina Torlak is our first customer. She used Lean 3, and she loves the system.

We talk to countless potential users.

Soonho has many ideas: Lean weekly meetings, slack forums, summaries with all success stories, …

We create LinkedIn accounts and ask Sebastian to create one too.



Leonardo de Moura • You

Senior Principal Applied Scientist at AWS, and Chief Architect at...

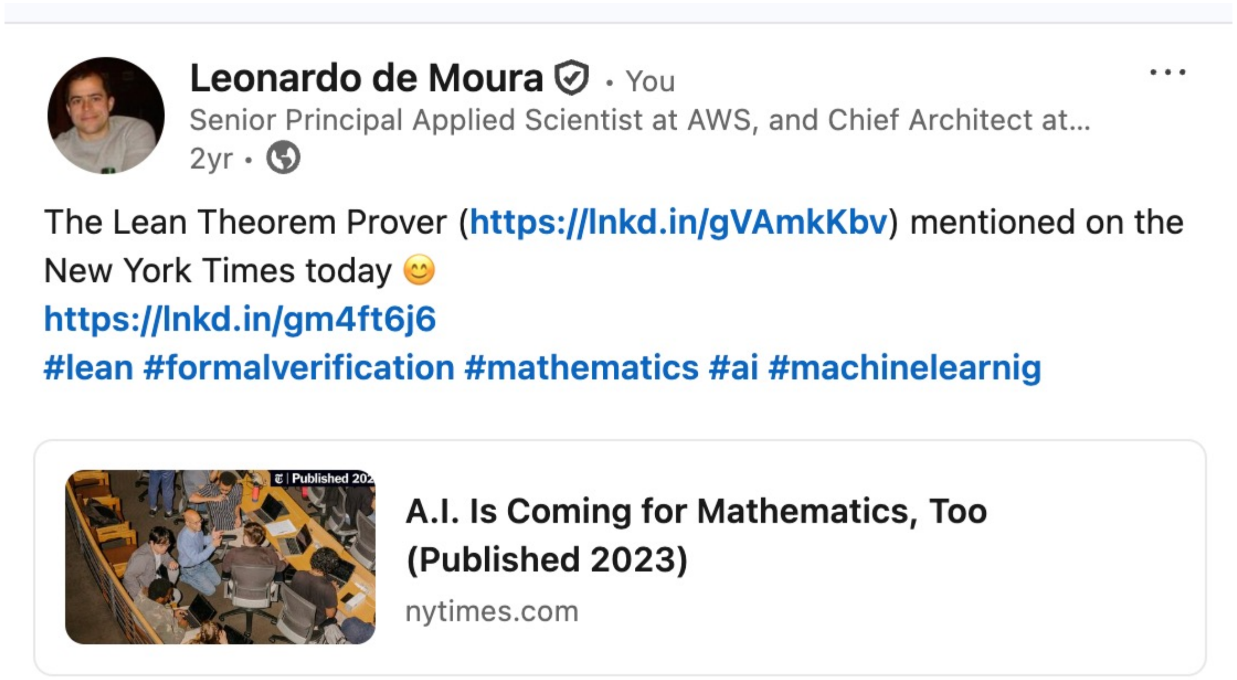
2yr •

I am thrilled to announce that the Mathlib (https://lnkd.in/gx6eh4aG) port to Lean 4 has been successfully completed this weekend. It is truly remarkable that over 1 million lines of formal mathematics have been successfully migrated. Once again, the community has amazed me and surpassed all my expectations. This achievement also aligns with the 10th anniversary of my initial commit to Lean on July 15, 2013. Patrick Massot has graciously shared a delightful video commemorating this significant milestone, which can be viewed here: https://lnkd.in/gjVr72t8.

Lean 4 overview for Mathlib users - Patrick Massot

youtube.com

Lean is on the NYT


Leonardo de Moura · You
Senior Principal Applied Scientist at AWS, and Chief Architect at...
2yr ·

The Lean Theorem Prover (https://lnkd.in/gVAmkKbv) mentioned on the New York Times today 😊
https://lnkd.in/gm4ft6j6
#lean #formalverification #mathematics #ai #machinelearnig

A.I. Is Coming for Mathematics, Too (Published 2023)
nytimes.com

The end of the pre-FRO era

Sebastian and I launch the Lean FRO a few days later.

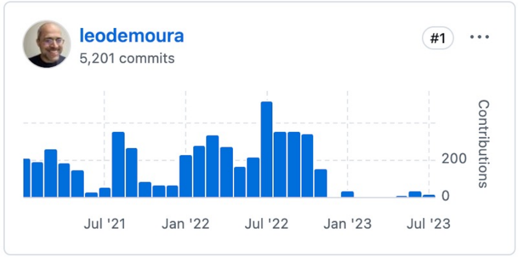
We have funding for a **5-year mission** to accelerate Lean's development.

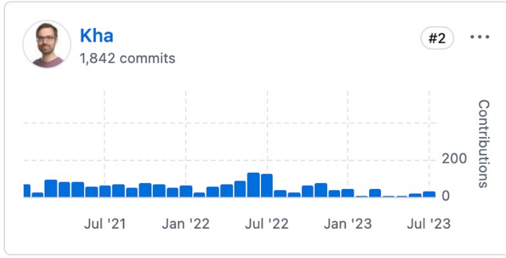
We finally have a team for developing Lean.

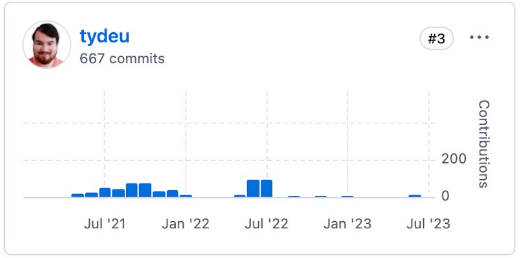
Lean is not a research project anymore.

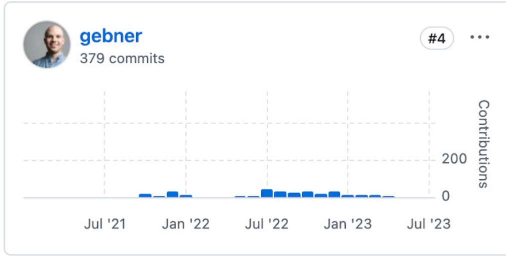
We run the Lean FRO as a startup.

Convergent Research









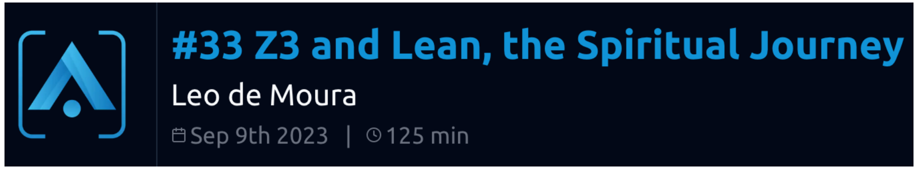
Lean FRO

Kim, David, Mac, Marc, Kyle all joined us at the Lean FRO.

Joachim Breitner joins us too.

Our roadmap is public.

From now on, it is a tsunami of success stories.

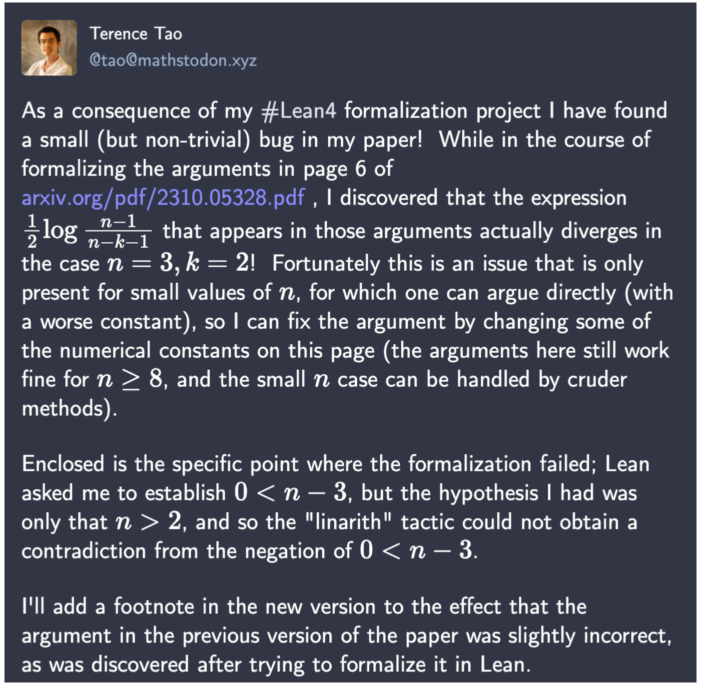


#33 Z3 and Lean, the Spiritual Journey

Leo de Moura

Sep 9th 2023 | 125 min

Terry Tao starts using Lean

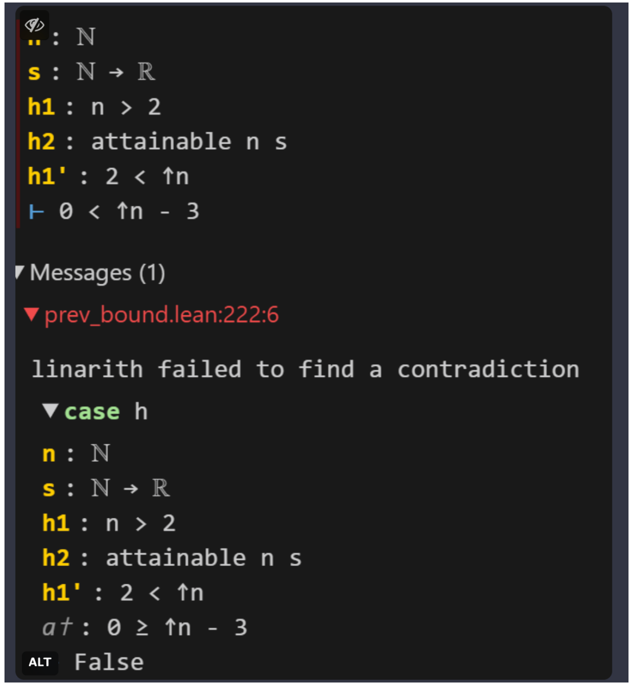


Terence Tao
@tao@mathstodon.xyz

As a consequence of my #Lean4 formalization project I have found a small (but non-trivial) bug in my paper! While in the course of formalizing the arguments in page 6 of arxiv.org/pdf/2310.05328.pdf , I discovered that the expression $\frac{1}{2}\log\frac{n-1}{n-k-1}$ that appears in those arguments actually diverges in the case $n = 3, k = 2$! Fortunately this is an issue that is only present for small values of n, for which one can argue directly (with a worse constant), so I can fix the argument by changing some of the numerical constants on this page (the arguments here still work fine for $n \geq 8$, and the small n case can be handled by cruder methods).

Enclosed is the specific point where the formalization failed; Lean asked me to establish $0 < n - 3$, but the hypothesis I had was only that $n > 2$, and so the "linarith" tactic could not obtain a contradiction from the negation of $0 < n - 3$.

I'll add a footnote in the new version to the effect that the argument in the previous version of the paper was slightly incorrect, as was discovered after trying to formalize it in Lean.



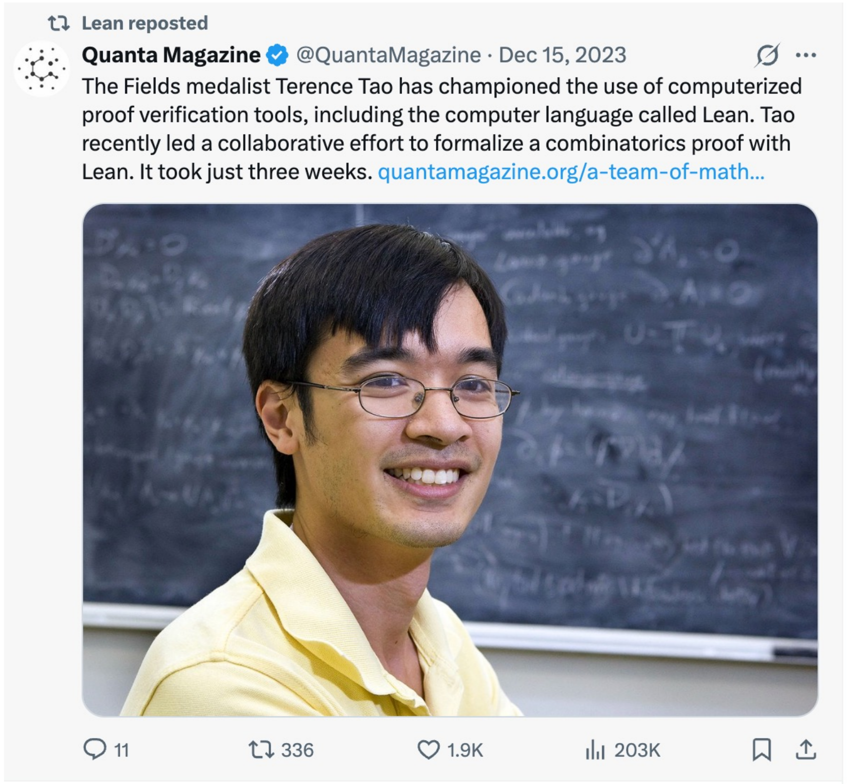
```
  : ℕ
s : ℕ → ℝ
h1 : n > 2
h2 : attainable n s
h1' : 2 < ↑n
⊢ 0 < ↑n - 3
```

▼ Messages (1)

▼ prev_bound.lean:222:6

```
linarith failed to find a contradiction
 ▼ case h
 n : ℕ
 s : ℕ → ℝ
 h1 : n > 2
 h2 : attainable n s
 h1' : 2 < ↑n
 a† : 0 ≥ ↑n - 3
 ⊢ False
```

And, a few months later...



Lean reposted

Quanta Magazine @QuantaMagazine · Dec 15, 2023

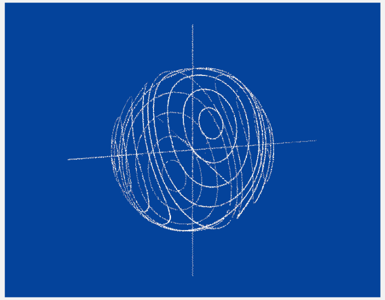
The Fields medalist Terence Tao has championed the use of computerized proof verification tools, including the computer language called Lean. Tao recently led a collaborative effort to formalize a combinatorics proof with Lean. It took just three weeks. quantamagazine.org/a-team-of-math...

11 336 1.9K 203K

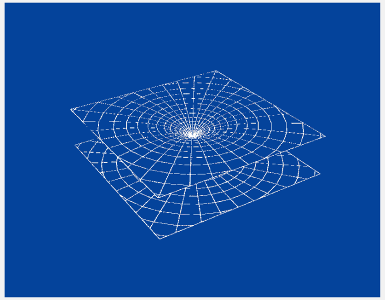
Harmonic: Mathematical Super Intelligence

Daniel Selsam introduces Tudor Achim and Vlad Tenev to me in 2023.

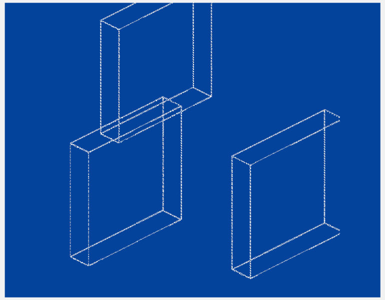
1 Building AI that is truthful, with verifiably correct and interpretable outputs



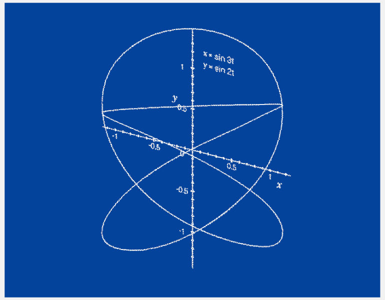
2 Changing the way mathematics is learned and taught in schools



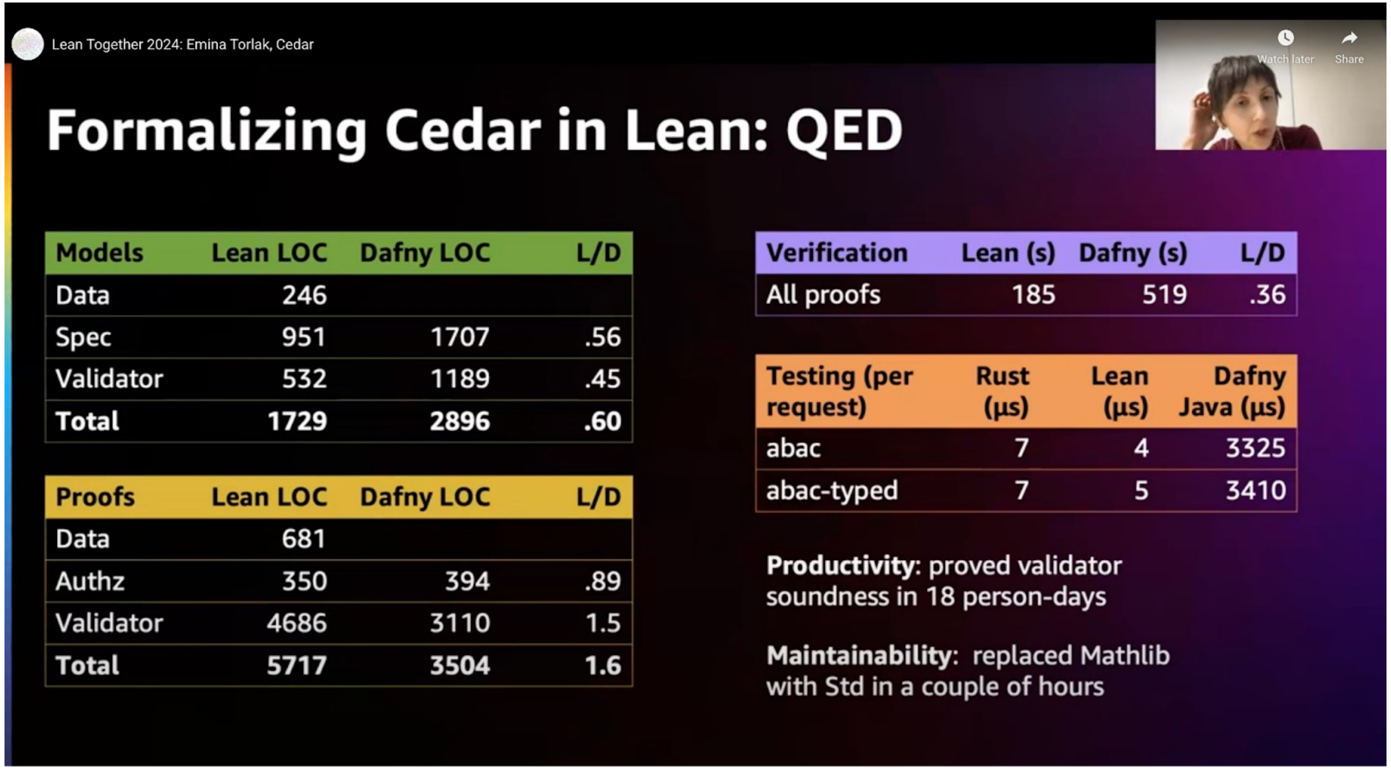
3 Accelerating the advent of verified software synthesis in safety-critical domains



4 Solving open problems in mathematics, science, and beyond



Cedar in Lean, the first success story at AWS


Lean Together 2024: Emina Torlak, Cedar

Watch later

Share

Formalizing Cedar in Lean: QED

Models	Lean LOC	Dafny LOC	L/D
Data	246		
Spec	951	1707	.56
Validator	532	1189	.45
Total	**1729**	**2896**	**.60**

Proofs	Lean LOC	Dafny LOC	L/D
Data	681		
Authz	350	394	.89
Validator	4686	3110	1.5
Total	**5717**	**3504**	**1.6**

Verification	Lean (s)	Dafny (s)	L/D
All proofs	185	519	.36

Testing (per request)	Rust (µs)	Lean (µs)	Dafny Java (µs)
abac	7	4	3325
abac-typed	7	5	3410

Productivity: proved validator soundness in 18 person-days

Maintainability: replaced Mathlib with Std in a couple of hours

SampCert

A project led by **Jean-Baptiste Tristan** at AWS.

An **open-source** Lean library of formally **verified differential privacy primitives**.

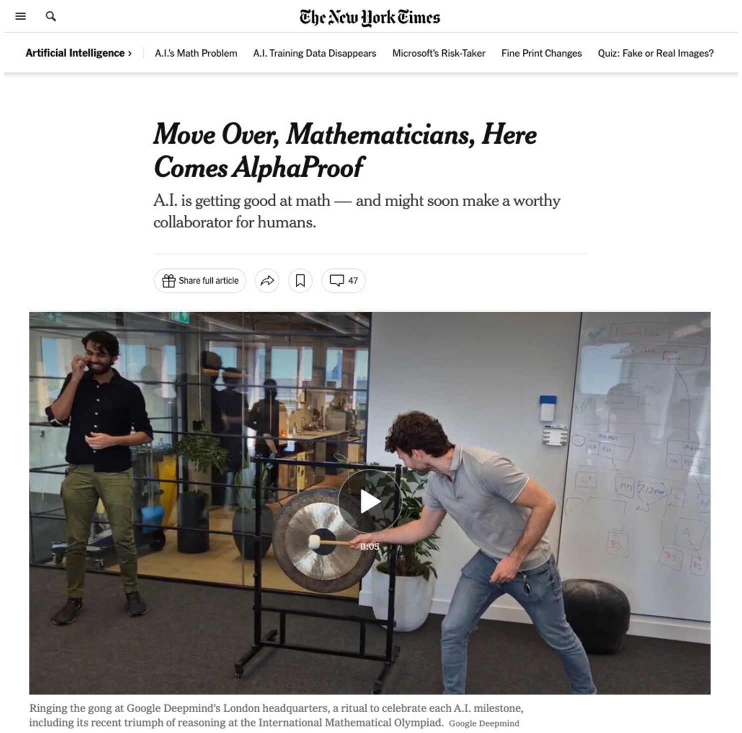
Tristan's implementation is not only verified, but it is also **twice as fast as the previous one**.

He managed to implement **aggressive optimizations** because Lean served as a guide, ensuring that **no bugs** were introduced.

SampCert is software, but its verification relies heavily on Mathlib.

The verification of code addressing practical problems in data privacy depends on the formalization of mathematical concepts, from **Fourier analysis** to **number theory** and **topology**.

“SampCert would not exist without Mathlib” – Jean-Baptiste Tristan



The New York Times

Artificial Intelligence › | A.I.'s Math Problem | A.I. Training Data Disappears | Microsoft's Risk-Taker | Fine Print Changes | Quiz: Fake or Real Images?

Move Over, Mathematicians, Here Comes AlphaProof

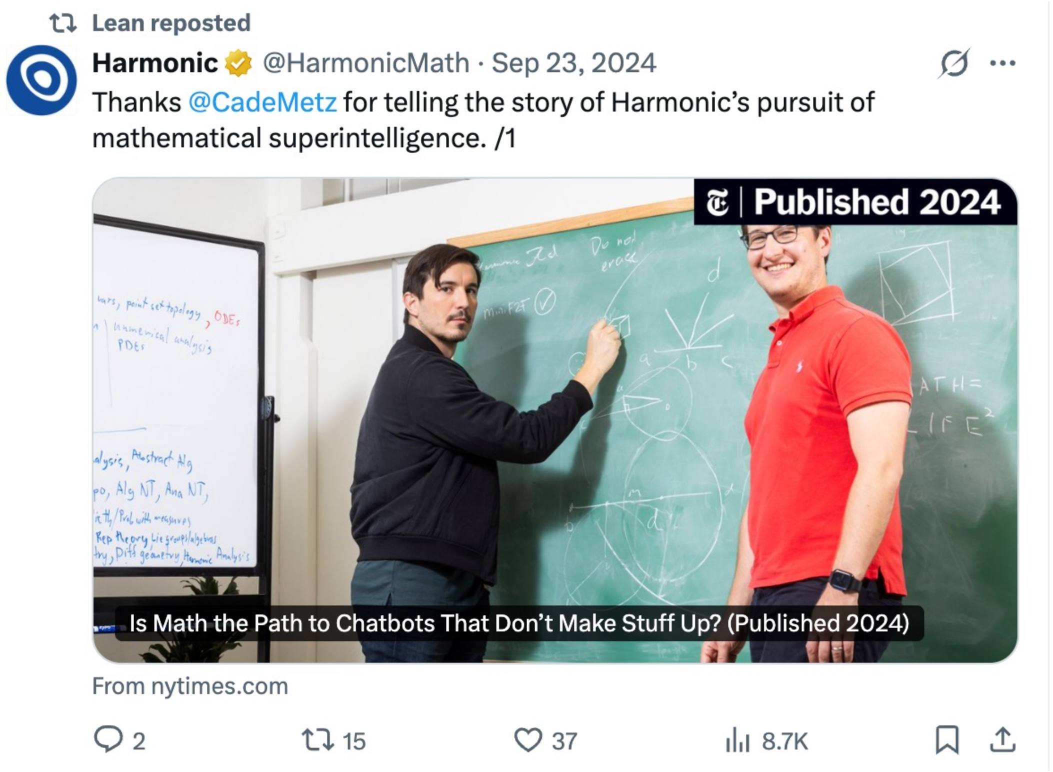
A.I. is getting good at math — and might soon make a worthy collaborator for humans.

Share full article | 47

Ringing the gong at Google Deepmind's London headquarters, a ritual to celebrate each A.I. milestone, including its recent triumph of reasoning at the International Mathematical Olympiad. Google Deepmind

"At Google DeepMind, we used Lean to build AlphaProof, a new reinforcement-learning based system for formal math reasoning ***. Lean's extensibility and verification capabilities were key in enabling the development of AlphaProof*** *." — Pushmeet Kohli, Vice President, Research Google DeepMind*

A few months later, Lean is back at the NYT



Lean reposted

Harmonic @HarmonicMath · Sep 23, 2024

Thanks @CadeMetz for telling the story of Harmonic's pursuit of mathematical superintelligence. /1

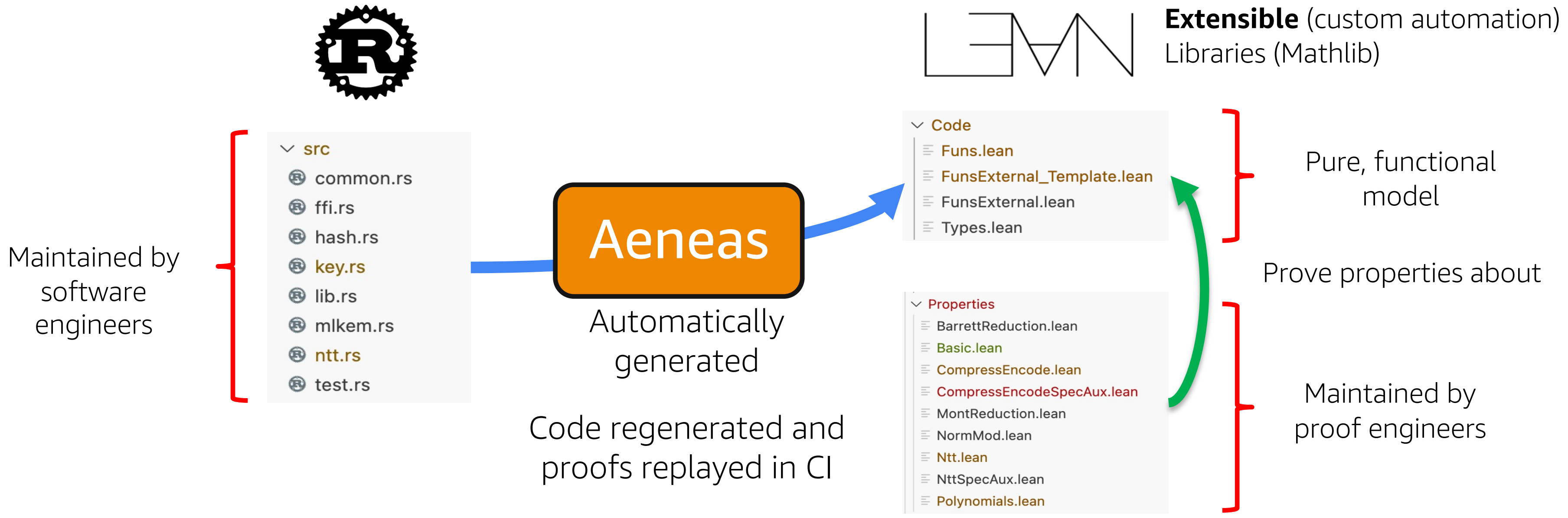
Published 2024

Is Math the Path to Chatbots That Don't Make Stuff Up? (Published 2024)

From nytimes.com

2 15 37 8.7K

Verifying Cryptography with Aeneas at Microsoft


Maintained by software engineers
Aeneas
Automatically generated
Code regenerated and proofs replayed in CI
Extensible (custom automation)
Libraries (Mathlib)
Pure, functional model
Prove properties about
Maintained by proof engineers

src
- common.rs
- ffi.rs
- hash.rs
- key.rs
- lib.rs
- mlkem.rs
- ntt.rs
- test.rs

Code
- Funs.lean
- FunsExternal_Template.lean
- FunsExternal.lean
- Types.lean

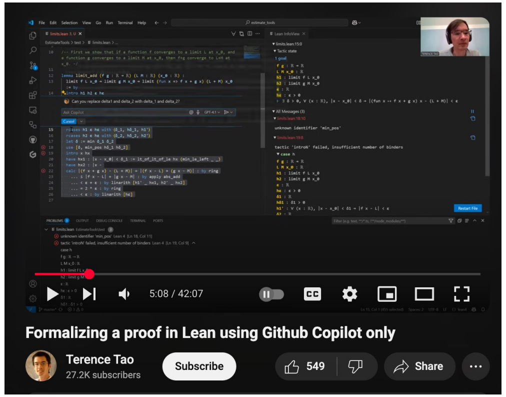
Properties
- BarrettReduction.lean
- Basic.lean
- CompressEncode.lean
- CompressEncodeSpecAux.lean
- MontReduction.lean
- NormMod.lean
- Ntt.lean
- NttSpecAux.lean
- Polynomials.lean

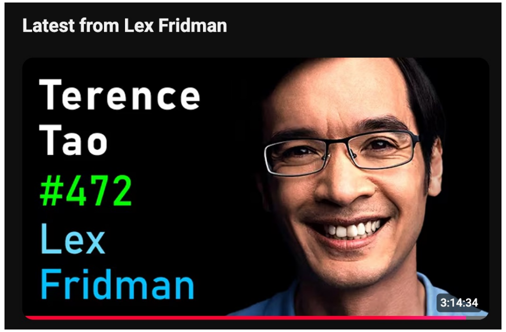
Lean is Taking Mathematics by Storm

*"**Lean enables large-scale collaboration** by allowing mathematicians to break down complex proofs into smaller, verifiable components. This formalization process ensures the correctness of proofs and facilitates contributions from a broader community. **With Lean, we are beginning to see how AI can accelerate the formalization of mathematics, opening up new possibilities for research.**" — Terence Tao*

Fermat's Last Theorem – Kevin Buzzard

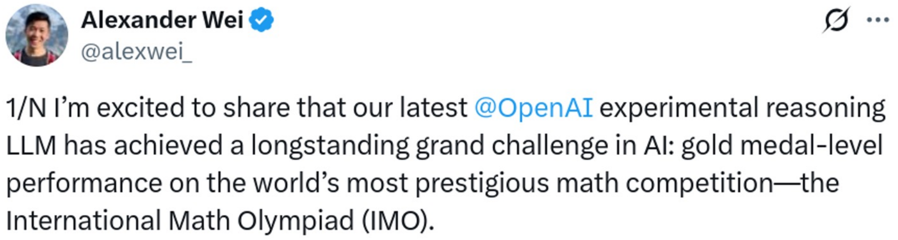
Carleson's Theorem (completed) – Floris van Doorn





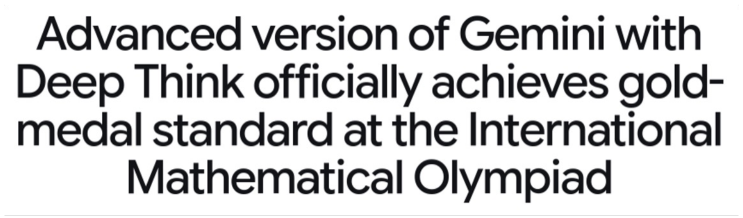
IMO 2025: 3 Gold, 1 Silver*

OpenAI (informal)

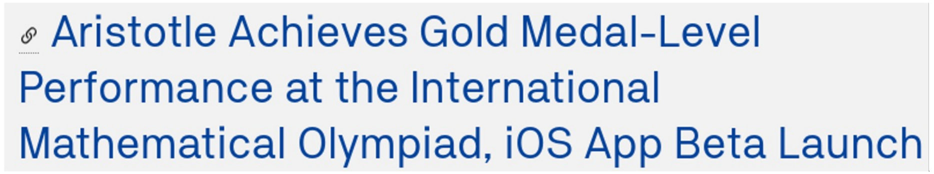

Alexander Wei
@alexwei_

1/N I’m excited to share that our latest @OpenAI experimental reasoning LLM has achieved a longstanding grand challenge in AI: gold medal-level performance on the world’s most prestigious math competition—the International Math Olympiad (IMO).

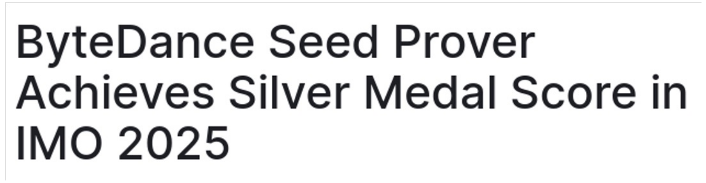
DeepMind (informal)


Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad

Harmonic (Lean)


Aristotle Achieves Gold Medal-Level Performance at the International Mathematical Olympiad, iOS App Beta Launch

ByteDance (Lean)


ByteDance Seed Prover Achieves Silver Medal Score in IMO 2025

Startups using Lean & AI

Harmonic

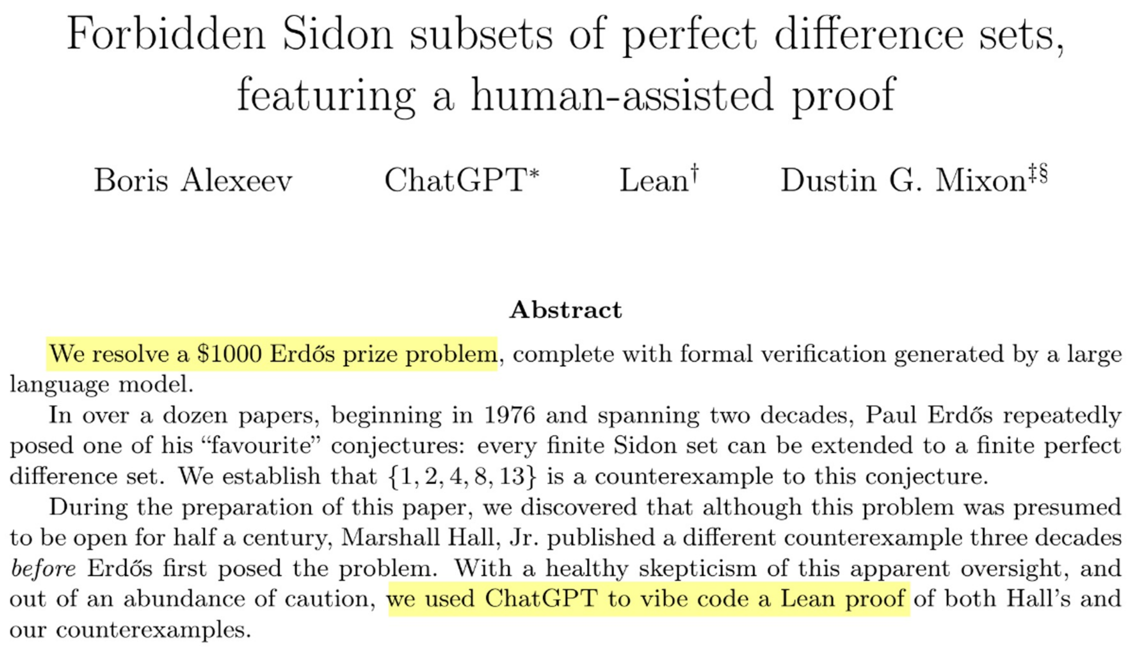
Axiom

Math, Inc.

Logical Intelligence

…and more to come

Vibe Proving



Forbidden Sidon subsets of perfect difference sets, featuring a human-assisted proof

Boris Alexeev ChatGPT* Lean† Dustin G. Mixon‡§

Abstract

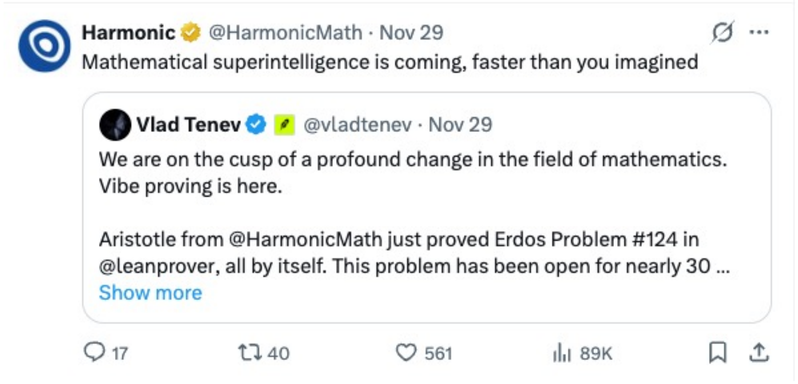
We resolve a $1000 Erdős prize problem, complete with formal verification generated by a large language model.

In over a dozen papers, beginning in 1976 and spanning two decades, Paul Erdős repeatedly posed one of his "favourite" conjectures: every finite Sidon set can be extended to a finite perfect difference set. We establish that $\{1, 2, 4, 8, 13\}$ is a counterexample to this conjecture.

During the preparation of this paper, we discovered that although this problem was presumed to be open for half a century, Marshall Hall, Jr. published a different counterexample three decades *before* Erdős first posed the problem. With a healthy skepticism of this apparent oversight, and out of an abundance of caution, we used ChatGPT to vibe code a Lean proof of both Hall's and our counterexamples.

borisalexeev.com/pdf/erdos707.pdf

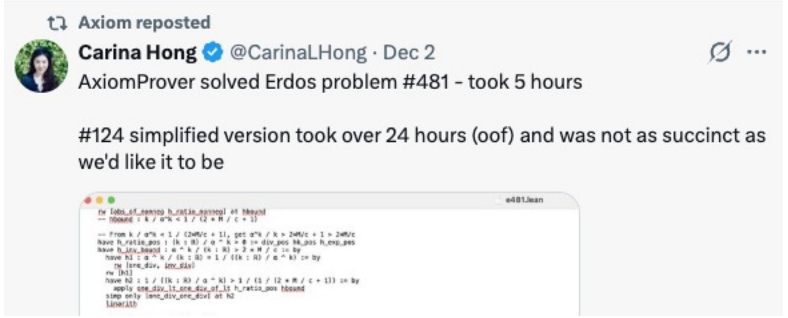
Vibe Proving


Harmonic @HarmonicMath · Nov 29
Mathematical superintelligence is coming, faster than you imagined

Vlad Tenev @vladtenev · Nov 29
We are on the cusp of a profound change in the field of mathematics. Vibe proving is here.

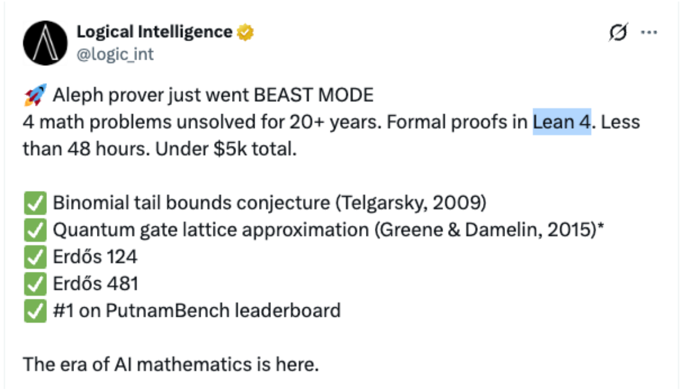
Aristotle from @HarmonicMath just proved Erdos Problem #124 in @leanprover, all by itself. This problem has been open for nearly 30 ...
Show more

17 40 561 89K


Axiom reposted
Carina Hong @CarinaLHong · Dec 2
AxiomProver solved Erdos problem #481 - took 5 hours

#124 simplified version took over 24 hours (oof) and was not as succinct as we'd like it to be

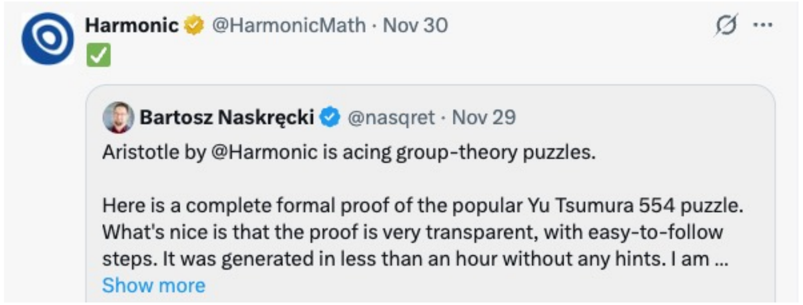
e481.lean
```
rw [abs_of_nonneg h_ratio_nonneg] at hbound
-- hbound : k / α^k < 1 / (2 * M / c + 1)

-- From k / α^k < 1 / (2*M/c + 1), get α^k / k > 2*M/c + 1 > 2*M/c
have h_ratio_pos : (k : ℝ) / α ^ k > 0 := div_pos hk_pos h_exp_pos
have h_inv_bound : α ^ k / (k : ℝ) > 2 * M / c := by
  have h1 : α ^ k / (k : ℝ) = 1 / ((k : ℝ) / α ^ k) := by
    rw [one_div, inv_div]
  rw [h1]
  have h2 : 1 / ((k : ℝ) / α ^ k) > 1 / (1 / (2 * M / c + 1)) := by
    apply one_div_lt_one_div_of_lt h_ratio_pos hbound
  simp only [one_div_one_div] at h2
  linarith
```


Logical Intelligence
@logic_int

🚀 Aleph prover just went BEAST MODE
4 math problems unsolved for 20+ years. Formal proofs in Lean 4. Less than 48 hours. Under $5k total.

✅ Binomial tail bounds conjecture (Telgarsky, 2009)
✅ Quantum gate lattice approximation (Greene & Damelin, 2015)*
✅ Erdős 124
✅ Erdős 481
✅ #1 on PutnamBench leaderboard

The era of AI mathematics is here.


Harmonic @HarmonicMath · Nov 30
✅

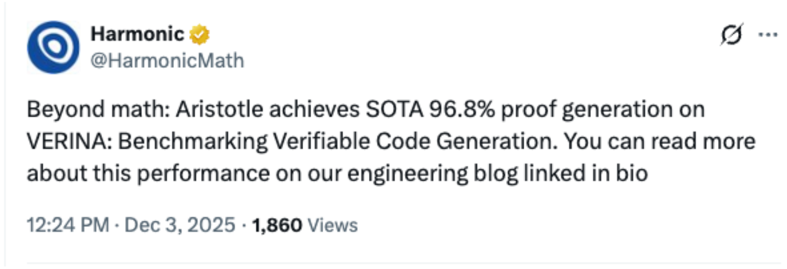
Bartosz Naskręcki @nasqret · Nov 29
Aristotle by @Harmonic is acing group-theory puzzles.

Here is a complete formal proof of the popular Yu Tsumura 554 puzzle. What's nice is that the proof is very transparent, with easy-to-follow steps. It was generated in less than an hour without any hints. I am ...
Show more

Prospective: AI+CS as the Next Frontier is Imminent

leanprover/cslib founded, supported by AWS, Google DeepMind, SDU, Centaur (Stanford)

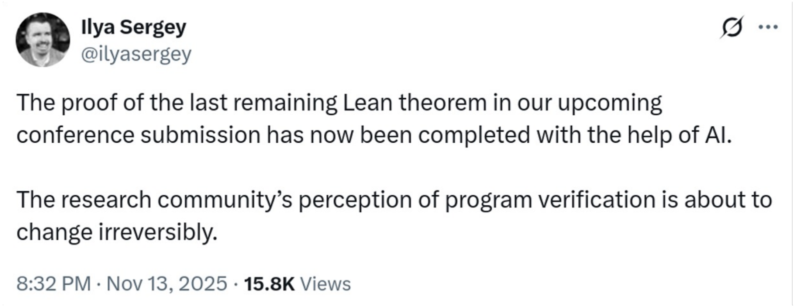
Companies are starting to pivot their math-proven AIs towards program verification



Harmonic
@HarmonicMath

Beyond math: Aristotle achieves SOTA 96.8% proof generation on VERINA: Benchmarking Verifiable Code Generation. You can read more about this performance on our engineering blog linked in bio

12:24 PM · Dec 3, 2025 · **1,860** Views



Ilya Sergey
@ilyasergey

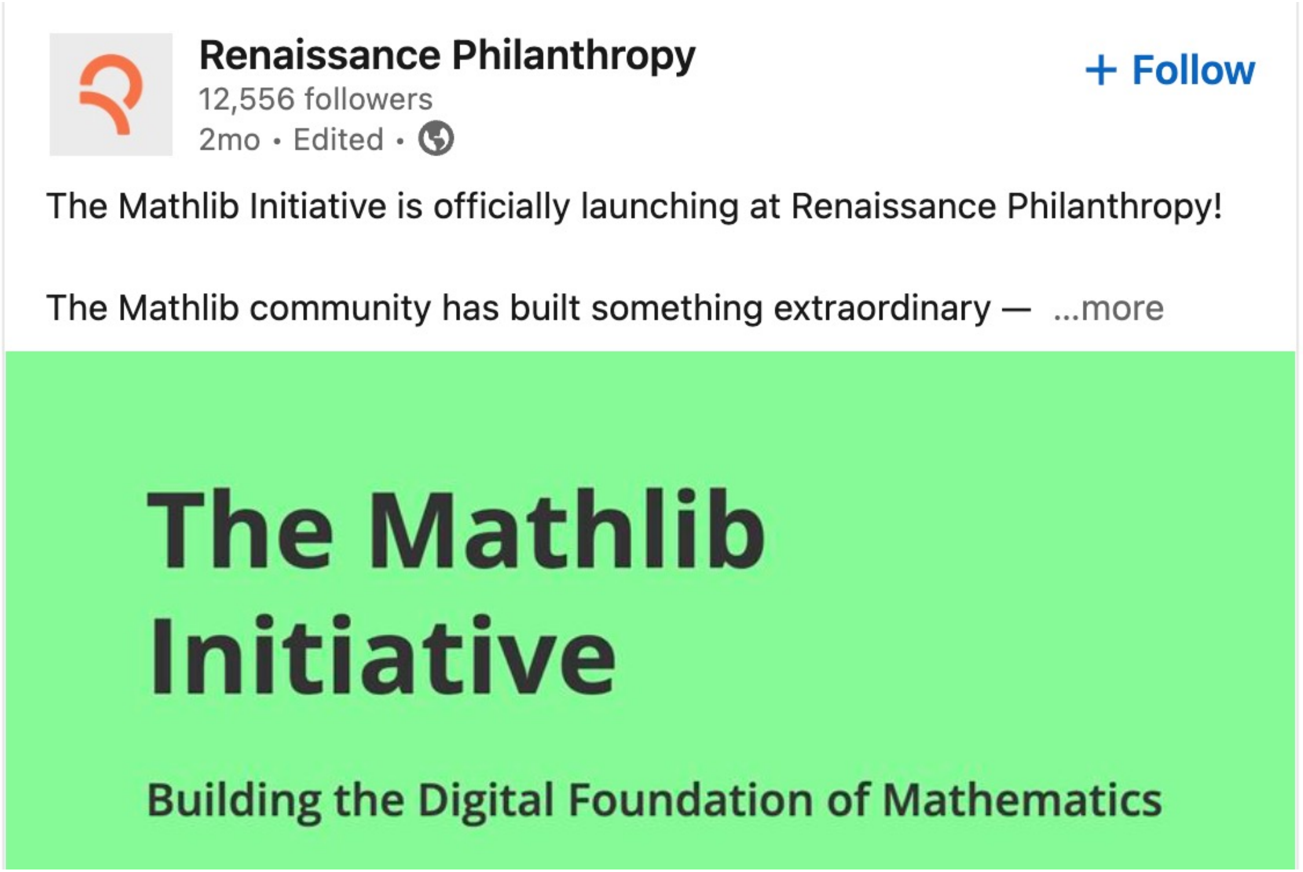
The proof of the last remaining Lean theorem in our upcoming conference submission has now been completed with the help of AI.

The research community's perception of program verification is about to change irreversibly.

8:32 PM · Nov 13, 2025 · **15.8K** Views

harmonic.fun/news#blog-post-aristotle-tech-report

Spec auto-formalization + program synthesis + verification as a service


Renaissance Philanthropy
12,556 followers
2mo • Edited •

+ Follow

The Mathlib Initiative is officially launching at Renaissance Philanthropy!

The Mathlib community has built something extraordinary — ...more

The Mathlib Initiative

Building the Digital Foundation of Mathematics

Lean FRO: by numbers

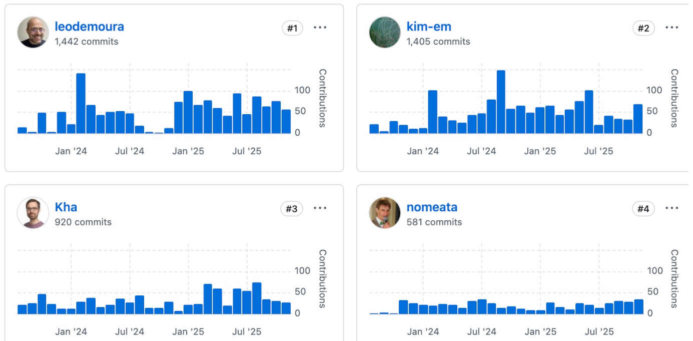
25 releases and **+7,000 pull requests** merged in the main repository only since its launch in July 2023.

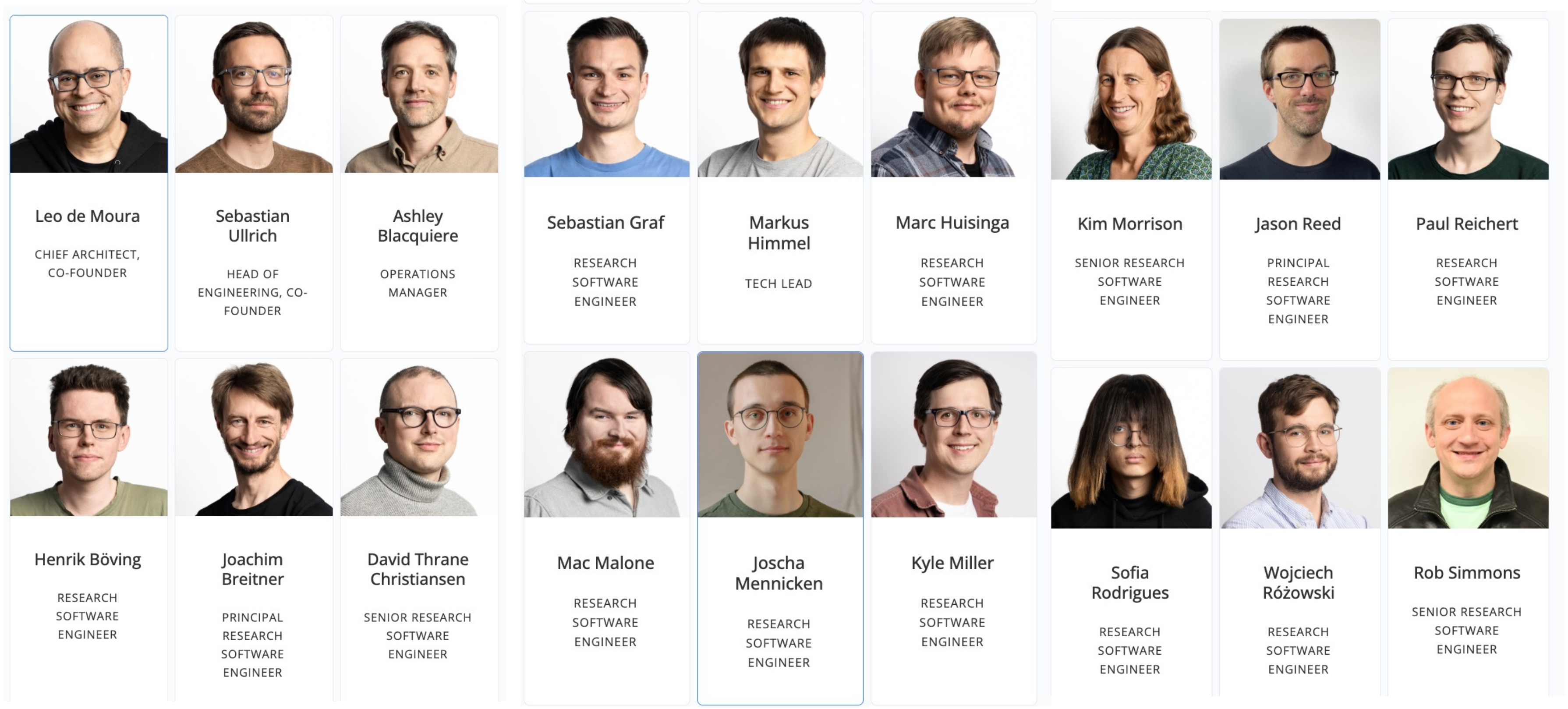
The Lean FRO has 19 people.

Public roadmaps: https://lean-lang.org/fro/roadmap/y3/

The Lean CADE 2015 paper received the Skolem (test of time) award in 2025.

Lean received the Programming Languages Software Award from ACM SIGPLAN in 2025 at PLDI exactly 10 years after Z3.





Leo de Moura
CHIEF ARCHITECT, CO-FOUNDER

Sebastian Ullrich
HEAD OF ENGINEERING, CO-FOUNDER

Ashley Blacquiere
OPERATIONS MANAGER

Henrik Böving
RESEARCH SOFTWARE ENGINEER

Joachim Breitner
PRINCIPAL RESEARCH SOFTWARE ENGINEER

David Thrane Christiansen
SENIOR RESEARCH SOFTWARE ENGINEER

Sebastian Graf
RESEARCH SOFTWARE ENGINEER

Markus Himmel
TECH LEAD

Marc Huisinga
RESEARCH SOFTWARE ENGINEER

Mac Malone
RESEARCH SOFTWARE ENGINEER

Joscha Mennicken
RESEARCH SOFTWARE ENGINEER

Kyle Miller
RESEARCH SOFTWARE ENGINEER

Kim Morrison
SENIOR RESEARCH SOFTWARE ENGINEER

Jason Reed
PRINCIPAL RESEARCH SOFTWARE ENGINEER

Paul Reichert
RESEARCH SOFTWARE ENGINEER

Sofia Rodrigues
RESEARCH SOFTWARE ENGINEER

Wojciech Różowski
RESEARCH SOFTWARE ENGINEER

Rob Simmons
SENIOR RESEARCH SOFTWARE ENGINEER

And Emilio J. Gallego Arias

Acknowledgements (and Many Thanks)

Sebastian Ullrich and Soonho for joining me on this journey and being true friends.

Sebastian Ullrich for unwavering belief in the project and support through challenges and successes.

Jeremy for friendship, being our first user, providing valuable feedback, advocacy, and constant encouragement.

Daniel for friendship, proposing the IMO GC, and many interesting discussions.

Joachim for friendship, great leadership, and always being ready to help and listen.

To **Ashley**, **David**, **Henrik**, **Kim**, **Kyle**, **Jason**, **Joscha**, **Mac**, **Marc**, **Markus**, **Paul**, **Rob**, **Sebastian Graf**, **Sofia**, and **Wojciech** for joining the Lean FRO.

To the entire Lean community.

What is grind?

New proof automation (Lean v4.22 – released mid August) developed by Kim Morrison and myself.

A proof-automation tactic **inspired by modern SMT solvers**. Think of it as a **virtual whiteboard**:

- Discovers new equalities, inequalities, etc.
- Writes facts on the board and merges equivalent terms
- Multiple engines cooperate on the same workspace

Cooperating Engines:

- Congruence closure; E-matching; Constraint propagation; Guided case analysis
- Satellite theory solvers (linear integer arithmetic, commutative rings, linear arithmetic)

Supports dependent types, type-class system, and dependent pattern matching

grind in a single example

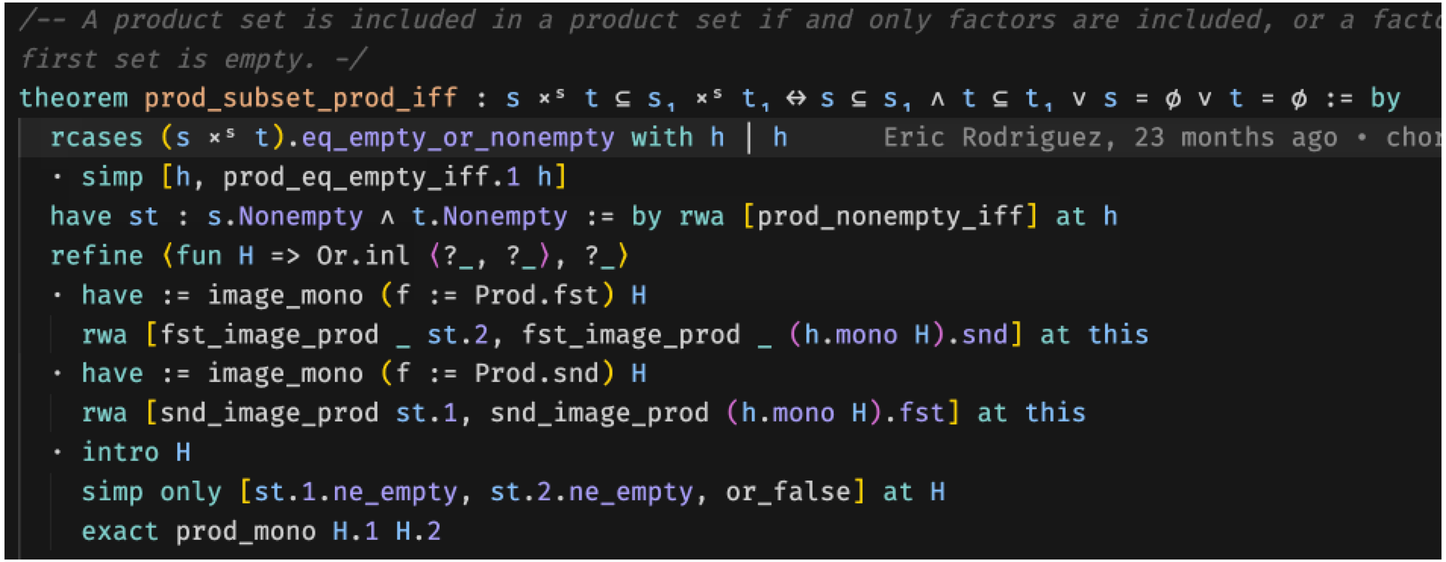
```
example {α} [CommRing α] [IsCharP α 0] (d t c : α) (d_inv PSO3_inv : α)
  (Δ40 : d^2 * (d + t - d * t - 2) * (d + t + d * t) = 0)
  (Δ41 : -d^4 * (d + t - d * t - 2) *
    (2 * d + 2 * d * t - 4 * d * t^2 + 2 * d * t^4 + 2 * d^2 * t^4 - c * (d + t + d * t)) = 0)
  : d * d_inv = 1 →
    (d + t - d * t - 2) * PSO3_inv = 1 →
    t^2 = t + 1 := by
grind
```

This is not a toy: it encodes a real algebraic constraint derived from relations among diagrams in a pivotal tensor category.

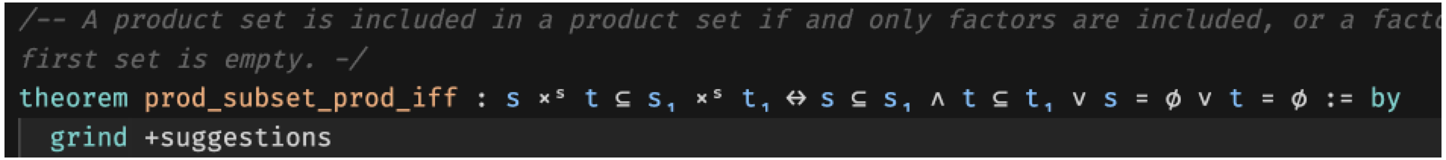
grind can handle this kind of reasoning automatically, in milliseconds.

Generate proof term is massive, more than 20,000 terms.

grind +suggestions in action

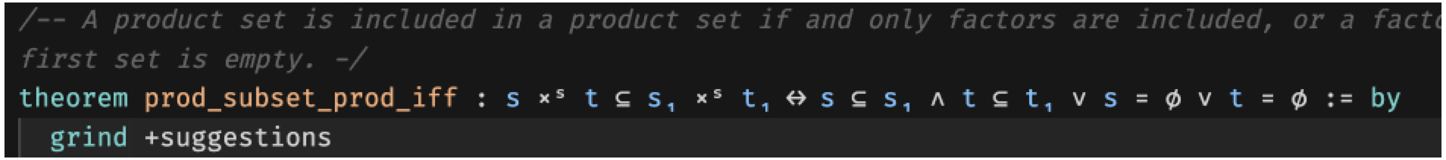


```
/-- A product set is included in a product set if and only factors are included, or a facto
first set is empty. -/
theorem prod_subset_prod_iff : s ×ˢ t ⊆ s₁ ×ˢ t₁ ↔ s ⊆ s₁ ∧ t ⊆ t₁ ∨ s = ∅ ∨ t = ∅ := by
  rcases (s ×ˢ t).eq_empty_or_nonempty with h | h      Eric Rodriguez, 23 months ago • chor
  · simp [h, prod_eq_empty_iff.1 h]
  have st : s.Nonempty ∧ t.Nonempty := by rwa [prod_nonempty_iff] at h
  refine ⟨fun H => Or.inl ⟨?_, ?_⟩, ?_⟩
  · have := image_mono (f := Prod.fst) H
    rwa [fst_image_prod _ st.2, fst_image_prod _ (h.mono H).snd] at this
  · have := image_mono (f := Prod.snd) H
    rwa [snd_image_prod st.1, snd_image_prod (h.mono H).fst] at this
  · intro H
    simp only [st.1.ne_empty, st.2.ne_empty, or_false] at H
    exact prod_mono H.1 H.2
```

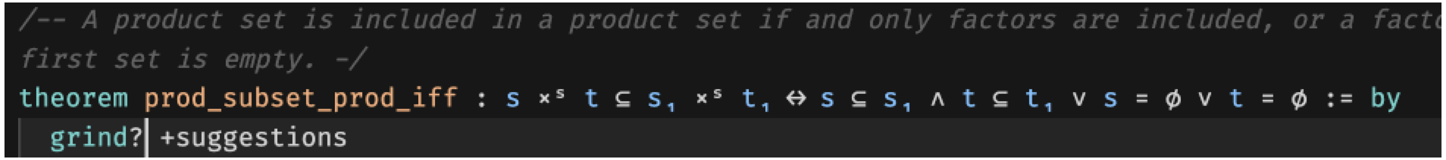


```
/-- A product set is included in a product set if and only factors are included, or a facto
first set is empty. -/
theorem prod_subset_prod_iff : s ×ˢ t ⊆ s₁ ×ˢ t₁ ↔ s ⊆ s₁ ∧ t ⊆ t₁ ∨ s = ∅ ∨ t = ∅ := by
  grind +suggestions
```

grind +suggestions: do you have to keep guessing?

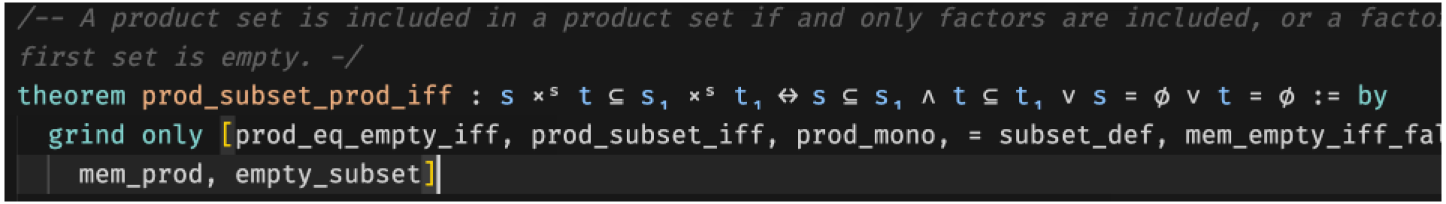


```
/-- A product set is included in a product set if and only factors are included, or a facto
first set is empty. -/
theorem prod_subset_prod_iff : s ×ˢ t ⊆ s₁ ×ˢ t₁ ↔ s ⊆ s₁ ∧ t ⊆ t₁ ∨ s = ∅ ∨ t = ∅ := by
  grind +suggestions
```



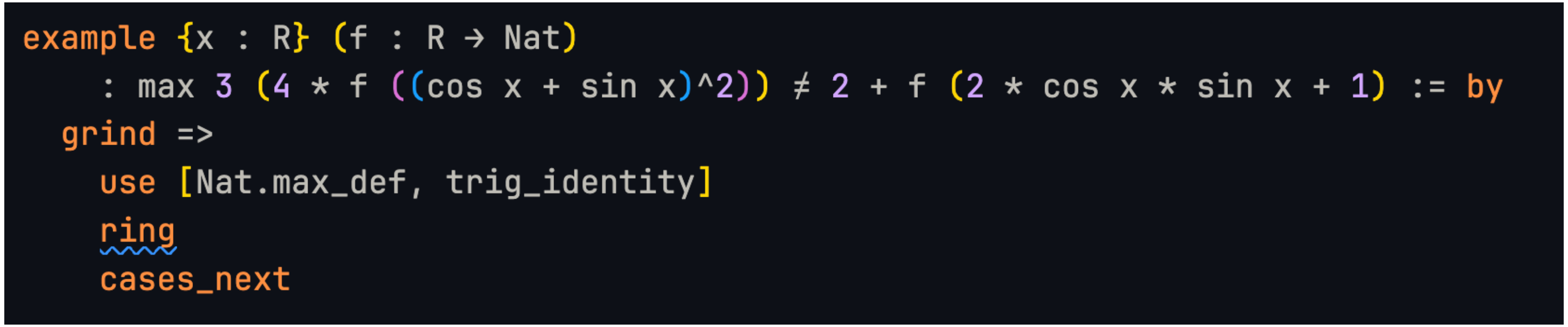
```
/-- A product set is included in a product set if and only factors are included, or a facto
first set is empty. -/
theorem prod_subset_prod_iff : s ×ˢ t ⊆ s₁ ×ˢ t₁ ↔ s ⊆ s₁ ∧ t ⊆ t₁ ∨ s = ∅ ∨ t = ∅ := by
  grind? +suggestions
```

grind finds out for you exactly with theorems are relevant



```
/-- A product set is included in a product set if and only factors are included, or a facto
first set is empty. -/
theorem prod_subset_prod_iff : s ×ˢ t ⊆ s₁ ×ˢ t₁ ↔ s ⊆ s₁ ∧ t ⊆ t₁ ∨ s = ∅ ∨ t = ∅ := by
  grind only [prod_eq_empty_iff, prod_subset_iff, prod_mono, = subset_def, mem_empty_iff_fal
    mem_prod, empty_subset]
```

grind interactive mode



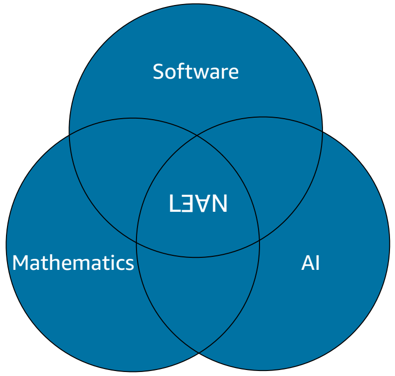
```lean
example {x : R} (f : R → Nat)
    : max 3 (4 * f ((cos x + sin x)^2)) ≠ 2 + f (2 * cos x * sin x + 1) := by
  grind =>
    use [Nat.max_def, trig_identity]
    ring
    cases_next
```

Already available in v4.25.0.

Higher level of abstraction.

Prediction: AI will learn how to use grind interactive mode in a few months.

Conclusion



Lean is an **efficient programming language** and **proof assistant**.

Lean is very extensible and is implemented in Lean.

Lean proofs are maintainable, stable, and transparent.

Progress is accelerating with the Lean FRO: module system, new compiler, new proof automation, etc.

The Mathlib community is changing how math is done.

AI + Lean will have a huge impact on mathematics and software/hardware verification.

It is not just about proving but also understanding complex objects and proofs, getting new insights, and navigating through the "thick jungles" that are **beyond our cognitive abilities**.

Thank You

https://leanprover.zulipchat.com/
x: @leanprover
LinkedIn: Lean FRO
Mastodon: @leanprover@functional.cafe
#leanlang, #leanprover

https://lean-lang.org/