

---

# AI & PROOFS

Flying Nobita

Feb 6, 2026

---

---

# ABOUT ME

Independent ZK Research Dev that works on ZK projects from Ethereum and its ecosystem.

Help run local communities in SG:

- ProgCryptoSG
- SoliditySG

---

# STORY

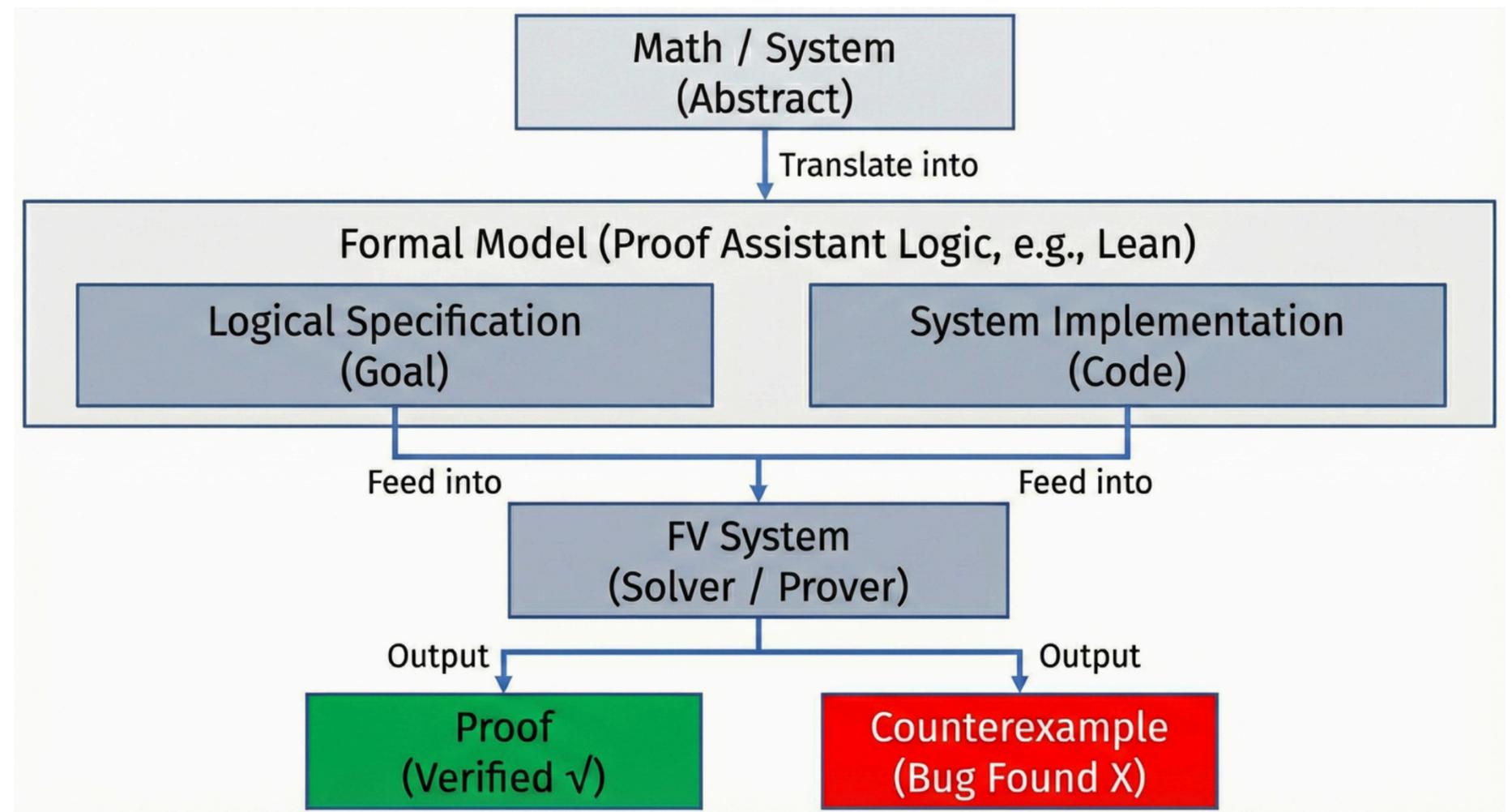
I went looking for tools to understand some mathematical concepts. I found a fascinating intersection of code and math. Here is my take on exploring AI-assisted proofs with an engineer's mindset.

---

# FORMAL VERIFICATION AND THEOREM PROVERS

Formal verification is a process of proving that a system satisfies its specification.

1. Write the system's specification in a proof assistant's logic (e.g. Lean)
2. Write the system's implementation also in Lean
3. Feed both into the FV System, which tries to prove the specification always holds for the implementation



---

# THEOREM PROVERS

A language that allows one to write formal proofs.

Theorem provers prevent AI hallucinations by using failed compilations as the ground truth.

Examples:

- Lean
- Coq
- Isabelle/HOL
- Acorn

---

# AI PROOF SYSTEM

An evolution of AI + Theorem Provers.

Submit math ideas in English and it formalizes them in Lean

Examples:

- Aristotle by Harmonic
- AlphaProof by Google DeepMind

# AI ASSISTED PROVING IS HERE



I solved my first Erdos Problem at 17!  
Thank you so much to everyone that helped me out along the way!

[erdosproblems.com/forum/thread/3...](https://erdosproblems.com/forum/thread/3...)

PROVED (LEAN)

Is there a sequence  $A = \{a_1 \leq a_2 \leq \dots\}$   
of integers with

$$\lim \frac{a_{n+1}}{a_n} = 2$$

such that

$$P(A') = \left\{ \sum_{n \in B} n : B \subseteq A' \text{ finite} \right\}$$

has density 1 for every cofinite  
subsequence  $A'$  of  $A$ ?

#347: [ErGr80]

number theory | complete sequences



A 17 y/o used AI for help to solve an Erdos problem  
"in the breaks between high-school classes".

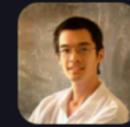
---

# V I B E P R O V I N G A N Y O N E ?

We all know about vibe coding, but what about **vibe proofing**?

Kevin Barreto, a math undergrad, used AI to solve an Erdős problem without knowing much about it.

---



Terence Tao

@tao@mathstodon.xyz

Recently, the application of AI tools to Erdos problems passed a milestone: an Erdos problem ([#728 erdosproblems.com/728](#)) was solved more or less autonomously by AI (after some feedback from an initial attempt), in the spirit of the problem (as reconstructed by the Erdos problem website community), with the result (to the best of our knowledge) not replicated in existing literature (although similar results proven by similar methods were located).

This is a demonstration of the genuine increase in capability of these tools in recent months, and is largely consistent with other recent demonstrations of AI using existing methods to resolve Erdos problems, although in most previous cases a solution to these problems was later located in the literature, as discussed in [mathstodon.xyz/deck/@tao/11578...](#) . This particular case was unusual in that the problem as stated by Erdos was misformulated, with a reconstruction of the problem in the intended spirit only obtained in the last few months, which helps explain the lack of prior literature on the problem. However, I would like to talk here about another aspect of the story which I find more interesting than the solution itself, which is the emerging AI-powered capability to rapidly write and rewrite expositions of the solution. (1/5)

---

# HOW TO START? WITHOUT (TOO MUCH) AI SLOP?

1. Upload the paper to LLM
2. Summarize the theorem in plain English
3. Ask LLM to write the formal proof in Lean
4. Compile Lean code and get error (hints)
5. Fix code based on the errors
6. Repeat 4-5 until proven

# A C O R N

- Provides a friendlier syntax than Lean.
- has an Assistant that gives you real-time feedback
- still in beta, with its math libraries still being built

The screenshot shows the Acorn Assistant interface. On the left is a code editor with a file named 'induction.ac' containing the following Lean code:

```
1 from nat import Nat
2 numerals Nat
3
4 define threeven(n: Nat) -> Bool {
5   | exists(d: Nat) {
6     | 3 * d = n
7   }
8 }
9
10 theorem zero_is_threeven {
11   | threeven(0)
12 }
13
14 theorem threeven_plus_three(n: Nat) {
15   | threeven(n) implies threeven(n + 3)
16 } by {
17   | let d: Nat satisfy {
18     | 3 * d = n
19   }
20   | 3 * (d + 1) = n + 3
21 }
22
23 define threeven_nearby(n: Nat) -> Bool {
24   | threeven(n) or threeven(n + 1) or threeven(n + 2)
25 }
26
27 theorem base_case {
28   | threeven_nearby(0)
29 }
30
```

On the right is the Acorn Assistant window, which displays the goal `exists(k0: Nat) { 3 * k0 = n }`. Below the goal, it states "The detailed proof has 2 steps:" and provides a table of proof steps:

Statement	Reason
<code>g9_0(T6_0, g6_3, s1_0(c1)) != c1(internal)</code>	negating the goal
<code>not g1_0(c1) or g9_0(T6_0, g6_3, s1_0(c1)) = c1(internal)</code>	the 'threeven' definition

# OTHER CHANNEL (TREAD CAREFULLY)

You can prove maths in Telegram:

The screenshot shows the Telegram channel interface for ClawHub. At the top, there is a navigation bar with the ClawHub logo, links for Skills, Upload, Import, Search, and Stars, and a user profile for @flyingnobita. The main content area features a post for 'Acorn Prover'. The post title is 'Acorn Prover'. The description reads: 'Verify and write proofs using the Acorn theorem prover for mathematical and cryptographic formalization. Use when working with Acorn proof files (.ac), verifying theorems, formalizing mathematical or cryptographic protocols, or writing proofs in the Acorn language. Triggers on: (1) Creating or editing .ac files, (2) Running acorn verify commands, (3) Formalizing math or crypto proofs, (4) Questions about Acorn syntax or standard library.' To the right of the description, there is a box indicating the 'CURRENT VERSION' is 'v1.0.0' and a red button labeled 'Download zip'. Below the description, there are statistics: '0 stars, 66 views, 0 current, 0 all-time' and the author '@flyingnobita'.

<https://clawhub.ai/flyingnobita/acorn-prover>

---

# TAKEAWAYS

1. Use AI to explain hard maths (obvious)
2. For best results, use ChatGPT 5.2 or Google Deep Thinking
3. Try to prove an unsolved Erdos problem for **fun**

---

**Q U E S T I O N S ?**

**T H A N K Y O U !**

---