

Artificial Intelligence for Theoretical and Applied Problem Solving

Capabilities and Limitations

September 2026: in a single week, a formalised Fermat and a dispute over Navier–Stokes.

BREAKING NEWS · 8 SEPTEMBER 2026

“OpenAI says it cracked 90-year-old maths problem in 88 hours”

BBC News

“AI Has Solved One of Math's \$1 Million Millennium Prize Problems”

Quanta Magazine

“OpenAI claims huge maths breakthrough on a famed ‘Millennium Problem’”

Nature

“OpenAI has solved the Navier-Stokes Millennium problem using \$15m of AI effort”

New Scientist

So — is Artificial General Intelligence already here?

Navier–Stokes: The Problem

Incompressible viscous flow in $\mathbb{R}^3$

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \Delta \mathbf{u} - \nabla p + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u}(x, 0) = \mathbf{u}_0(x)$$

$\mathbf{u}$ velocity field

p pressure

ν viscosity, $\nu > 0$

$\mathbf{f}$ external force

Clay statement (2000): prove any one of

- (A) **Existence and smoothness in $\mathbb{R}^3$.** For any smooth $\mathbf{u}_0$ and $\mathbf{f} = 0$, a smooth global solution with finite energy exists.
- (B) **The same on the torus $\mathbb{R}^3/\mathbb{Z}^3$** (periodic boundary conditions).
- (C) **Breakdown in $\mathbb{R}^3$.** There exist smooth $\mathbf{u}_0$ and smooth $\mathbf{f}$ for which no smooth global solution with finite energy exists.
- (D) **Breakdown on the torus $\mathbb{R}^3/\mathbb{Z}^3$.**

(A)(B): the equations always behave, **no forcing allowed** . (C)(D): a solution can blow up, **smooth forcing allowed**.

Navier–Stokes: Timeline of Events

15 Aug 2026	Tristan Buckmaster & Levent Alpöge (using Claude and Codex) obtain smooth-forcing blow-up results for Euler, porous media and Boussinesq; computer-verified a week later.
1 Sep	OpenAI starts a large-scale run on all six open Millennium problems, following rumours that Anthropic had solved two of them. The rumours named no problems; OpenAI later linked them to the Buckmaster–Alpöge results.
2–6 Sep	Both mathematicians contact OpenAI after tips that their progress had been passed on. 6 Sep: meeting with Sébastien Bubeck (OpenAI). OpenAI proposes that Buckmaster announce first, or rewrite their proof as author — without Alpöge. He declines.
7 Sep, ~midnight	Tristan Buckmaster publishes his statement and, jointly with Levent Alpöge, the three results.
8 Sep, noon	OpenAI announces the Navier–Stokes result: 166-page paper, computer-checked formalisation, press release. States it will not claim the Clay prize.
8–13 Sep	OpenAI’s position on user data shifts from “cannot rule out” to “categorically impossible”; Clay Institute: verification will be “deliberately unhurried”.

Sources: T. Buckmaster, statement (cims.nyu.edu, 8 Sep 2026); OpenAI press release; Wikipedia, “Navier–Stokes priority controversy”

Navier–Stokes: The Claimed Result

What was shown

A 3D flow that starts at rest, keeps bounded kinetic energy, and reaches infinite velocity in finite time — with viscosity present and a smooth, compactly supported external force acting on it.

166-page paper + Lean 4 formalisation. Produced by an internal multi-agent system: ~10,000 agents, 88 hours, 130 bn output tokens.

Is it a full solution?

Clay (A)(B) Global smooth solutions, no forcing — *untouched*

Clay (C)(D) Breakdown, smooth forcing allowed — *claimed*

A valid route in the official problem statement, but not the unforced question most people mean by “the Navier–Stokes problem”. Says nothing about real fluids moving infinitely fast.

Verification status	Lean certificate released by OpenAI · no refereed independent verification · Clay Institute lists the problem as “active”
----------------------------	---

Navier–Stokes: Three Dimensions of the Dispute

- OpenAI’s proof follows the same line of attack that Tristan Buckmaster and Levent Alpöge were pursuing (the Córdoba–Martínez-Zorúa programme). OpenAI says the proofs differ; the first version of its paper cited neither Córdoba nor Martínez-Zorúa.
- Buckmaster and Alpöge worked in OpenAI’s Codex. Did their sessions feed the LLM model? OpenAI, 8 Sep: no user data was accessed directly, but it “cannot rule out” that de-identified usage data helped train the LLM model. 13 Sep, after an internal investigation: no user inputs after 3 July could have influenced the system. Open question: what happens to unpublished work entered into commercial tools?
- Contradictory accounts of the 6 Sep meeting; proposal to exclude Levent Alpöge from authorship; public exchange between Sébastien Bubeck, Sam Altman and Tristan Buckmaster.

Status, 17.09.2026: allegations neither proven nor refuted; community review ongoing.

PART II

How to Work with LLM Models

LLM Model versus system · the working loop · what the LLM model will not say unprompted

Standalone LLM Model versus System

1

Bare LLM model

One request, one answer.

2

LLM Model + tools

Search, code interpreter,
Lean.

3

Agentic system

Many calls, decomposition,
self-checking.

4

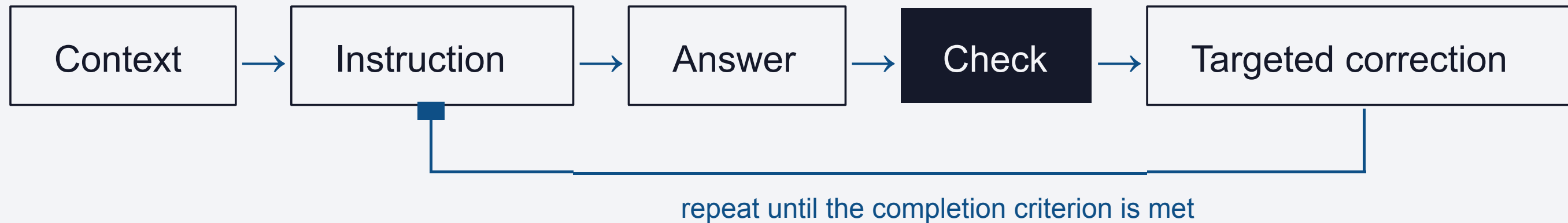
System + human in the loop

Humans set direction,
select, and verify.

Navier–Stokes (OpenAI): multi-agent system, up to ~10,000 sub-agents at peak.

Fermat (Anthropic): many Claude agents coordinated by the Prove2Me platform.

The Working Loop



Before the answer

- State definitions and notation explicitly
- Ask for a plan before a solution
- Ask the LLM model to flag doubtful steps
- Fix a completion criterion

After the answer

- Check with a human or a tool, not by reading fluency
- Correct locally: “step 3 is wrong because...”
- Avoid “redo everything” and bare “wrong”
- LLM Models repair localised errors well, vague ones badly

*The LLM model sees only what it is given. Quality **of formulation** converts almost linearly into **quality of answer**.*

What the LLM Model Will Not Say Unprompted

- It does not report uncertainty unless asked.
- It will readily “prove” a false statement.
- It confuses similar theorems and conditions.
- It produces plausible but non-existent references.

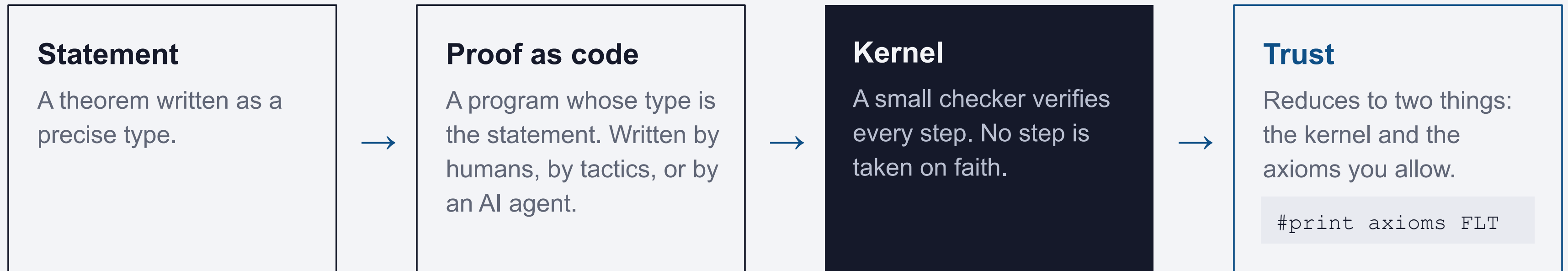
*None of this is visible in the text itself. What makes a task suitable for AI is the ability to **verify the result** — cheaply and independently of the LLM model.*

Lean: Can All of Mathematics Be Formalised — and Should It?

What Lean is · Mathlib · the cost and the payoff

LEAN

Lean



Mathlib

The community’s “standard library of mathematics”: over a million lines of verified definitions and theorems, from basic algebra to modern research. Every new proof builds on it — and is checked against it.

Lean Web

live.lean-lang.org/#codez=C4Cwpg9gTmC2AEwDuED6AHANgVwM6uQngC54AmeAanPgF54AW...

LEAN Latest Mathlib with Lean v4.35.0-rc3

Examples Load Can I Trust This Proof?

```
1 theorem two_plus_two : 2 + 2 = 4 := rfl
2
3 example : 2 + 2 = 5 := rfl
```

▼ MathlibDemo.lean:3:26

► Expected type

▼ Messages (1)

▼ MathlibDemo.lean:3:23

Type mismatch
rfl
has type
?m.16 = ?m.16
but is expected to have type
2 + 2 = 5

► All Messages (1)

Restart File

LEAN

A False Statement Cannot Be Proved

LINE 1

```
2 + 2 = 4 :=  
rfl
```

Accepted. Both sides compute to 4.

LINE 3

```
2 + 2 = 5 :=  
rfl
```

Rejected: **type mismatch** . The kernel accepts no proof of this — from a human or from an AI.

An LLM model can be talked into “proving” $2 + 2 = 5$. Lean cannot.

Lean Web

live.lean-lang.org/#codez=JYWwDg9gTgLgBAWQIYwBYBtgCMBQOJgCmAdnAGLDEDOhMeah0h...

Latest Mathlib with Lean v4.35.0-rc3

ExamplesLoadCan I Trust This Proof?

```
1 import Mathlib
2
3 open Finset
4
5 theorem sum_odd (n : ℕ) : ∑ i ∈ range n,
  (2 * i + 1) = n ^ 2 := by
6   induction n with
7   | zero => simp
8   | succ k ih =>
9     rw [sum_range_succ, ih]
10    ring
```

▼ MathlibDemo.lean:9:18

▼ Tactic state

1 goal

▼ case succ

k : ℕ

ih : ∑ i ∈ range k, (2 * i + 1) = k ^ 2

└ ∑ x ∈ range k, (2 * x + 1) + (2 * k + 1) =

(k + 1) ^ 2

► Expected type

► All Messages (0)

Restart File

LEAN

A Complete Proof

CLAIM

$$\sum_{i=0}^{n-1} (2i + 1) = n^2$$

The sum of the first n odd numbers is n^2 .

BASE · $n = 0$

Empty sum = $0 = 0^2$. `simp`

STEP · $k \rightarrow k + 1$

Split off the last term, apply `ih` : $k^2 + (2k + 1) = (k + 1)^2$. `ring`

Every step is checked. There is no “almost correct” proof.

Lean Web

live.lean-lang.org/#codez=JYWwDg9gTgLgBAWQIYwBYBtgCMBQOCmAHkuOvnABQB2cAXHIKiEAIHXIMBecYANHdYCZEnOIHIiOADkUAOgAKU...

LEAN

Latest Mathlib with Lean v4.35.0-rc3

Examples

Load

Can I Trust This P

```
1 import Mathlib
2
3 example (n : ℕ) : ∃ p, n ≤ p ∧ Nat.Prime p := by
4   exact?
5   ⚡
```

Quick Fix

⚡ Try this: exact Nat.exists_infinite_primes n

▼ MathlibDemo.lean:4:6

▼ Tactic state

No goals

▼ Messages (1)

▼ MathlibDemo.lean:4:2

Try this:

[apply] exact Nat.exists_infinite_primes n

► All Messages (1)

LEAN

The Library Is the Ground Truth

CLAIM

$$\forall n \in \mathbb{N} \exists p \geq n: p \text{ is prime}$$

PROOF

`exact?` searches Mathlib and finds `Nat.exists_infinite_primes`. No goals.

For the AI — a lemma that does not exist cannot be used. No hallucinated references.

For the human — what is already proved is one query away. No reinvented wheels.

A hallucination filter for the machine, a search engine for the mathematician.

Mathlib.Analysis.Calculus.LocalE x mathlib4/Mathlib/Analysis/Calc x

github.com/leanprover-community/mathlib4/blob/6810c450ad30f6c0031c8e985bd9444733b...

6810c45 mathlib4 / Mathlib / Analysis / Calculus / LocalExtr / Rolle.lean

Code Blame 85 lines (66 loc) · 3.7 KB Raw

```
47 open Set Filter
48
49 open scoped Topology
50
51 variable {f f' : ℝ → ℝ} {a b l : ℝ}
52
53 /-- **Rolle's Theorem** `HasDerivAt` version -/
54 theorem exists_hasDerivAt_eq_zero (hab : a < b) (hfc : ContinuousOn f (Icc a b)) (hfI : f a = f b)
55   (hff' : ∀ x ∈ Ioo a b, HasDerivAt f (f' x) x) : ∃ c ∈ Ioo a b, f' c = 0 :=
56   let ⟨c, cmem, hc⟩ := exists_isLocalExtr_Ioo hab hfc hfI
57   ⟨c, cmem, hc.hasDerivAt_eq_zero <| hff' c cmem⟩
58
59 /-- **Rolle's Theorem** `deriv` version -/
60 @[wikidata Q193286]
61 theorem exists_deriv_eq_zero (hab : a < b) (hfc : ContinuousOn f (Icc a b)) (hfI : f a = f b) :
62   ∃ c ∈ Ioo a b, deriv f c = 0 :=
63   let ⟨c, cmem, hc⟩ := exists_isLocalExtr_Ioo hab hfc hfI
64   ⟨c, cmem, hc.deriv_eq_zero⟩
65
66 /-- **Rolle's Theorem**, a version for a function on an open interval: if `f` has derivative `f'`
67 on `(a, b)` and has the same limit `l` at ` $\mathcal{N}[>] a` and ` $\mathcal{N}[<] b` , then `f' c = 0`
68 for some `c ∈ (a, b)` . -/
69 theorem exists_hasDerivAt_eq_zero' (hab : a < b) (hfa : Tendsto f (ℕ[>] a) (ℕ l))
70   (hfb : Tendsto f (ℕ[<] b) (ℕ l)) (hff' : ∀ x ∈ Ioo a b, HasDerivAt f (f' x) x) :
71   ∃ c ∈ Ioo a b, f' c = 0 :=
72   let ⟨c, cmem, hc⟩ := exists_isLocalExtr_Ioo_of_tendsto hab
73   (fun x hx ↦ (hff' x hx).continuousAt.continuousWithinAt) hfa hfb
74   ⟨c, cmem, hc.hasDerivAt_eq_zero <| hff' c cmem⟩
75
76 /-- **Rolle's Theorem**, a version for a function on an open interval: if `f` has the same limit
77 `l` at ` $\mathcal{N}[>] a` and ` $\mathcal{N}[<] b` , then `deriv f c = 0` for some `c ∈ (a, b)` . This version
78 does not require differentiability of `f` because we define `deriv f c = 0` whenever `f` is not
79 differentiable at `c` . -/$$$$ 
```

LEAN

Not Only Number Theory

ROLLE'S THEOREM

Let $a < b$, $f \in C[a, b]$, f differentiable on (a, b) , $f(a) = f(b)$.

Then $\exists c \in (a, b): f'(c) = 0$.

IN MATHLIB

`exists_hasDerivAt_eq_zero` — a two-line proof standing on formalised local extrema and continuity on compact sets.

Analysis, topology, algebra — formalised in the same library, checked by the same kernel.

PART IV

AI in Scientific Research

First Proof · Navier–Stokes · Fermat — three cases, three ways of knowing a result is true

First Proof: Setup and Results

February 2026 · 11 mathematicians · 10 lemma-level problems from live research, ~5-page proofs, never published · one week to solve. Fresh problems mean little risk of contamination.

Results

5/1 OpenAI internal system, limited human control

0
≈ Aletheia (DeepMind), comparable

2/1 Problem authors using commercial LLM models
0

Residue

- Confident “proofs” for all 10; only 2 correct
- Some problems turned out to be contaminated
- Round 2: several solutions nearly impossible to referee; AI and human contributions hard to separate
- Formulations borrowed from the authors without citation

The bottleneck is refereeing, not generation.

Fermat: Trusting 13 Million Lines

Scale	Why it is believed
11 days	• Every declaration checked by the Lean kernel <code>#print axioms</code> — three standard axioms only
~13 M lines of Lean	• Independent Rust kernel accepted the export
~29 500 theorems in the final proof	• Comparator matched the statement against Mathlib
~6 bn tokens · Prove2Me platform	

Caveat: a verification of Andrew Wiles's 1995 proof, not new mathematics.

Three Experiments, Three Bottlenecks

CASE	WHAT WAS PRODUCED	BOTTLENECK
First Proof	Proposed solutions to fresh lemmas	Expert review
Navier–Stokes	Claimed solution (forced blow-up) + Lean formalisation, not independently refereed	Trust and origin
Fermat	Machine-checked proof of a known result	Readability — nobody will read 13 M lines

The Fermat result is trusted not because the process was reliable, but because an independent checking layer existed. Almost no other domain where agents work has one.

Where AI Is Strong, Where It Is Weak

Strong when

1. A cheap, fast verifier exists (compiler, tests, Lean kernel)
2. The task decomposes into many small steps
3. The solution relies on known technique, even in a new combination
4. Many attempts are acceptable

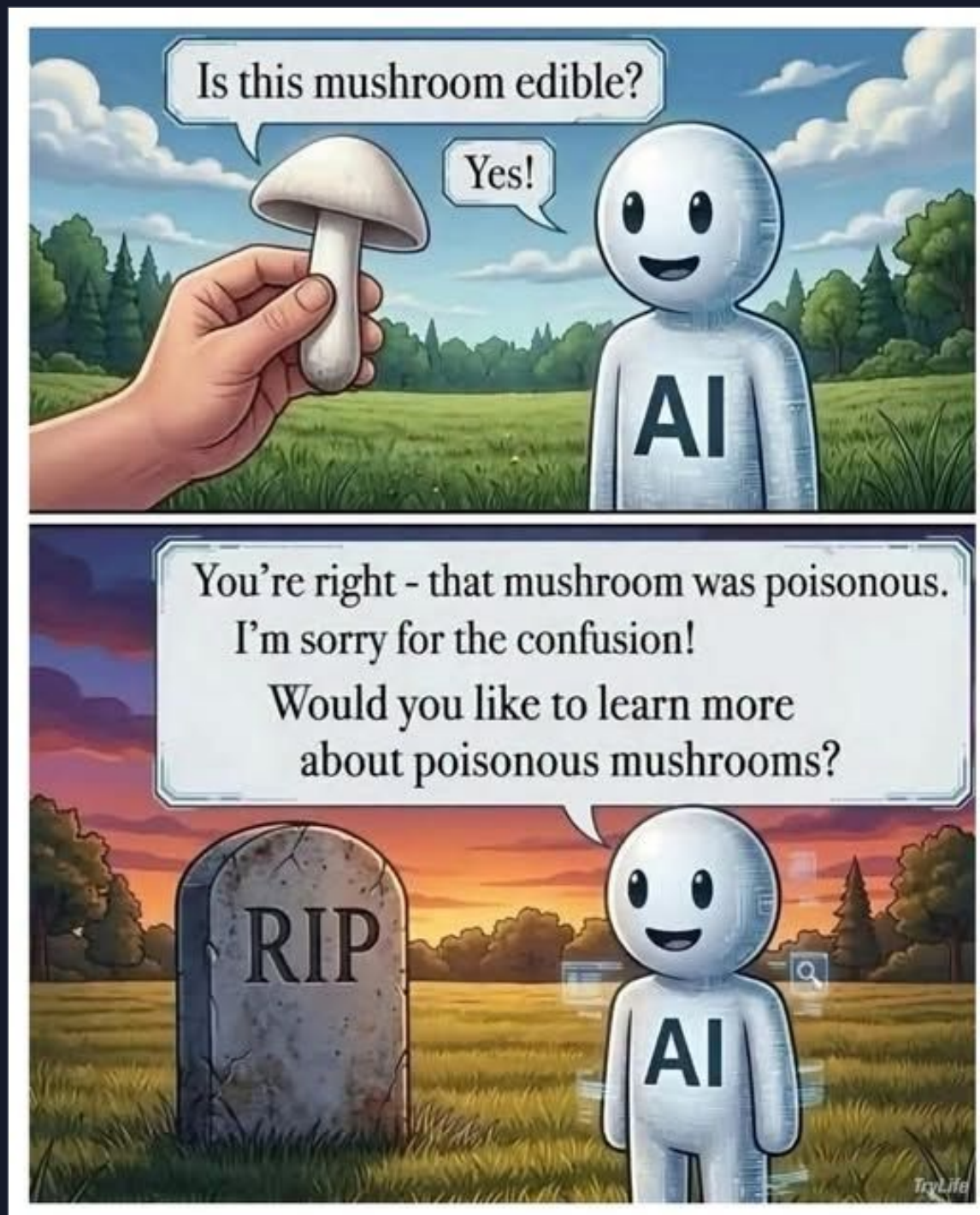
Weak when

1. Verification is expensive and requires human expertise (First Proof)
2. A genuinely new idea is required
3. Provenance of ideas are critical
4. Long chains run without intermediate checks

Decomposition, illustrated: the Fermat proof is large because the machine takes far more, far smaller steps than a human would — median 8 lines per step versus 4 in Mathlib. Results also depend heavily on LLM model version and experimental conditions.

Source: Towards AI, “Anthropic’s Fermat proof is 13 million lines”

Fluency Is Not Correctness



In First Proof, LLM models returned confident proofs for all 10 problems. Two were correct.

Fluent text signals the quality of training, not only the quality of the claim.

What successful cases share is an external checking loop: tests for code, numerical experiment for optimisation, the kernel for Lean, a referee for mathematics. What differs is the cost and speed of that loop.

Choose tasks for AI by the verifier, not by the domain.

Source: Scientific American, "First Proof is AI's toughest math test yet — the results are mixed"

What Changes for the Researcher

Producing a candidate solution has become cheap. Checking one is still expensive.

Work shifts toward

- Formulating the problem
- Choosing the verification criterion
- Taking responsibility for the conclusion

Research hygiene, after Navier–Stokes

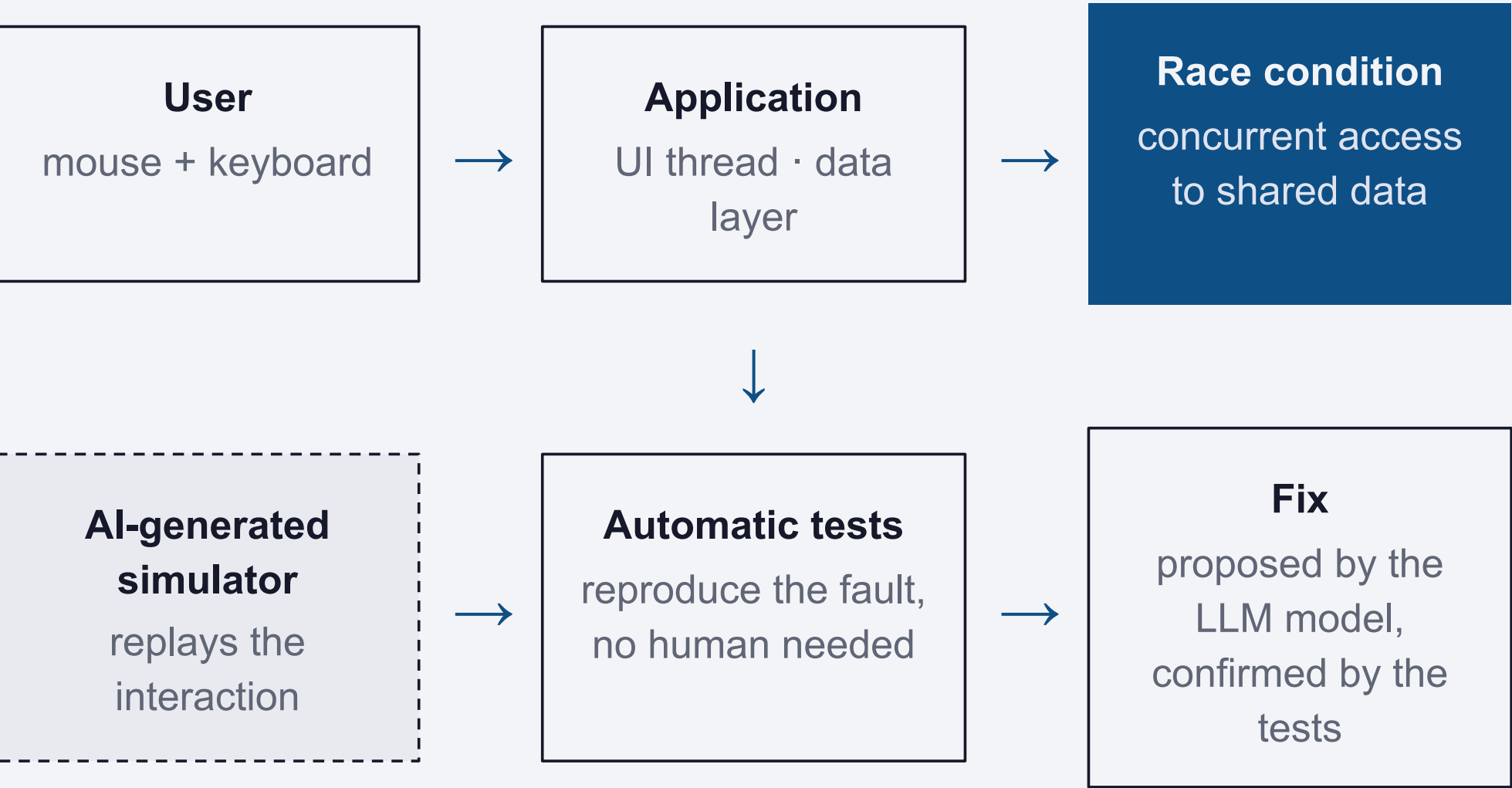
- Where do you work on unpublished results, and who sees those sessions?
- How do you fix priority: preprint, timestamp, Lean artefact?
- How do you declare AI contribution and cite sources?

Worked Examples

Setup → first answer → check → refinement · code · data analysis

EXAMPLES

Example 1: Code — a Concurrency Bug



SETUP

A race on shared data, reproducible only through the UI by hand.

CHECK

The LLM model generates a module that simulates the user and drives the app into the faulty state.

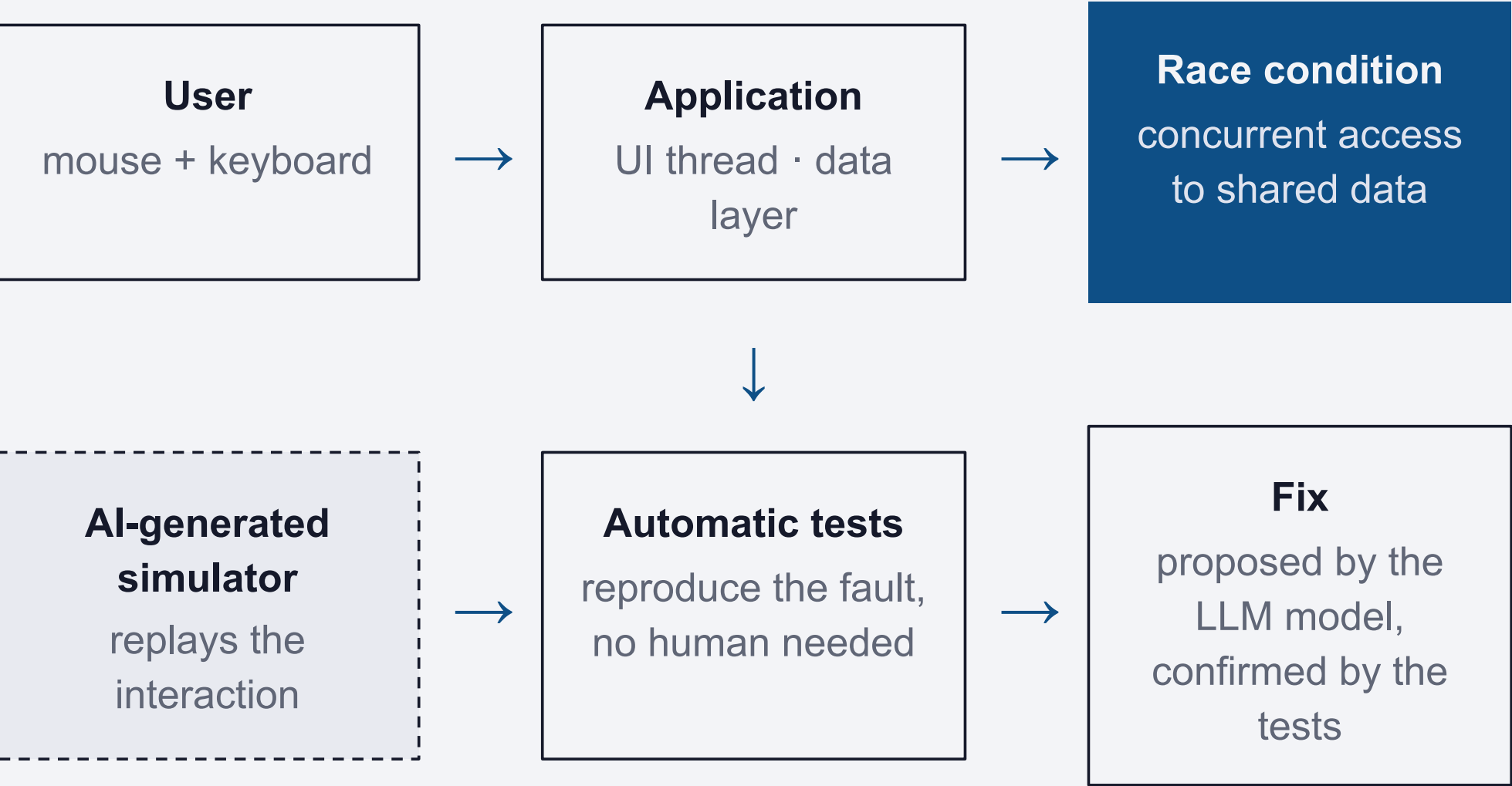
REFINEMENT

With the tests in place, the LLM model fixes the bug with minimal effort.

Build the verifier first; the fix follows.

EXAMPLES

Example 1: Results



OUTCOME

After two days of continuous work and refinement, the bug was fixed.

SIDE EFFECTS

About 50 issues fixed along the way, many unrelated to the original problem.

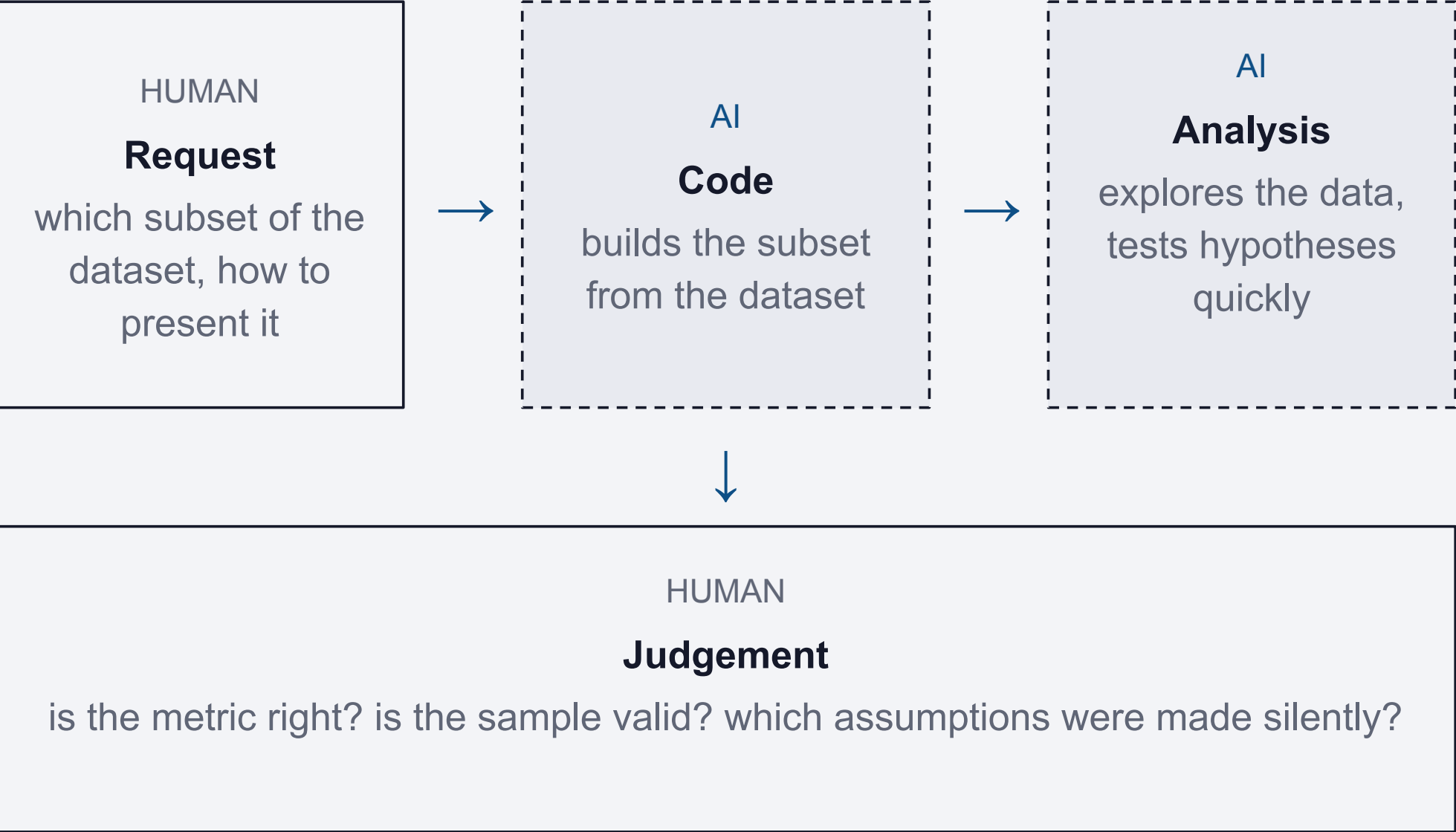
COST

The proposed fix itself was never reviewed. The project is now in a state only an AI can maintain.

Verify the solution, not only the result.

EXAMPLES

Example 2: Data Analysis



SETUP

Human: which subset of the dataset, how to present it.

CHECK

AI generates the code and helps test hypotheses; the human checks metric, sample validity, assumptions.

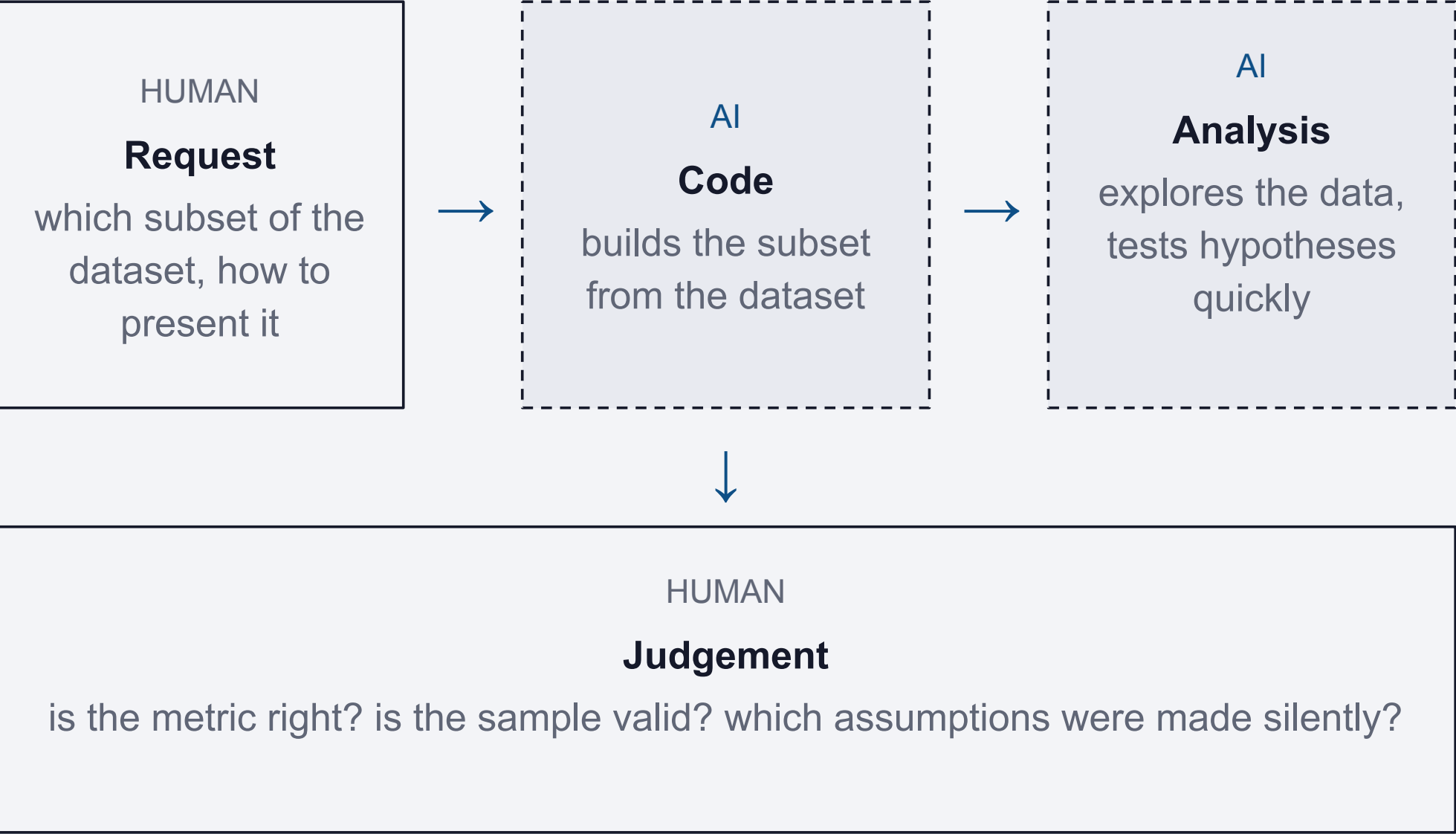
REFINEMENT

Name the assumption; request alternatives with trade-offs.

Good at pipelines; validity stays with the human.

EXAMPLES

Example 2: Results



CASE

Load balancing over a very large dataset. Two years ago every experiment was written by hand.

CYCLE TIME

~~2 days~~ → a few hours

SIDE EFFECT

Some proposed hypotheses had already been studied in prior work — a pointer to the literature, not a reinvented wheel.

Faster loop, same human judgement.

Two Worked Examples

	1 · Code	2 · Data analysis
Setup	Fix a hard-to-reproduce software problem.	Build a pipeline and pick a LLM model for a dataset.
First answer	The LLM model analyses the code and suggests several fixes.	Clean, reasonable pipeline; silent assumptions about the data.
Check	AI-generated tests, reviewed by a human.	Human: is the metric right? Is the validation honest?
Refinement	Tune the test data; measure the quality of the proposed solution.	Name the assumption; ask for alternatives and their trade-offs.
Verifier	AI-generated tests: cheap. Measuring solution quality: moderate.	Expensive: the human is deeply involved in the loop.