Formal Reasoning in AI

From next-step prediction to verifiable reasoning agents

Peiyang Song

PhD Student @ CMU LTI

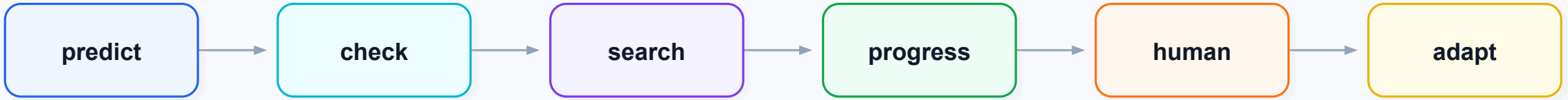
ECE 4424 / CS 4824 · Machine Learning
Virginia Tech · Sept. 14, 2026

PRELIMINARIES

Reliable reasoning is a system problem

Not just a better next-token predictor.

Strong reasoning emerges when learning is placed inside a loop with exact feedback, search, human control, and adaptation.

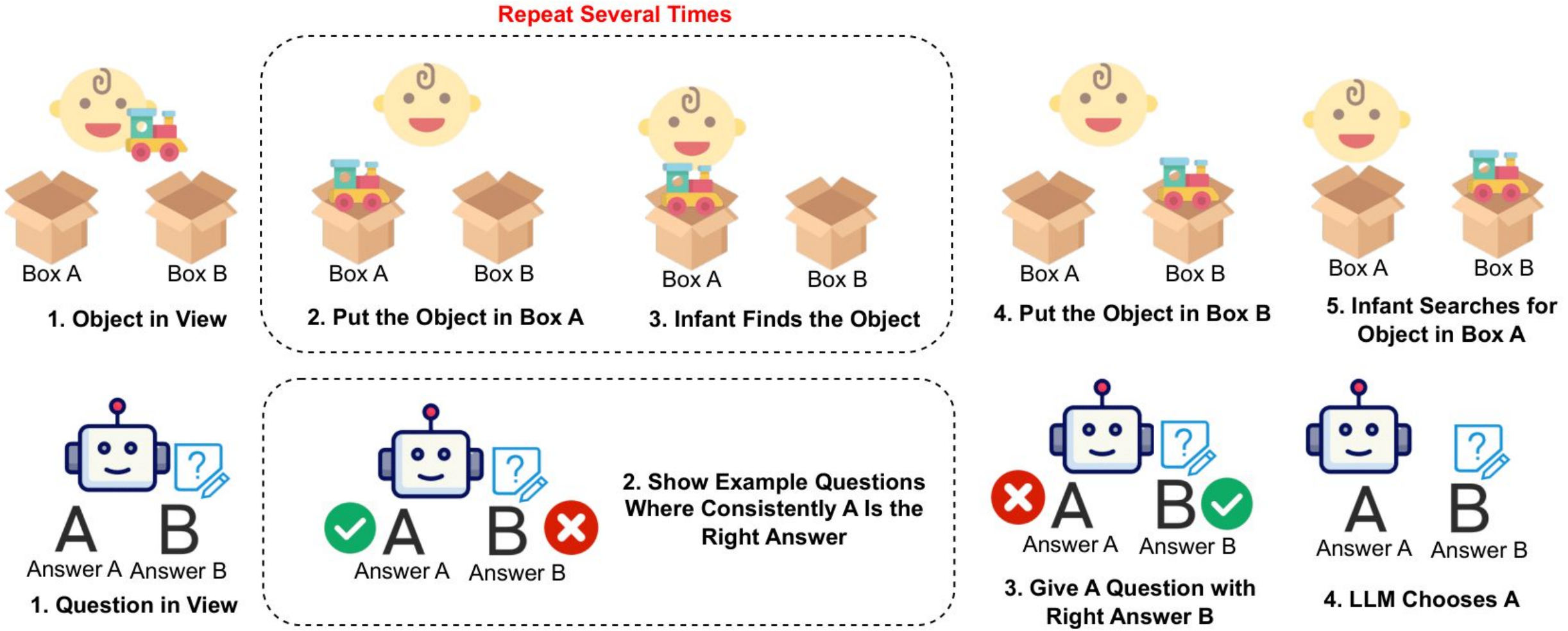


Today’s path: failure → formal feedback → neuro-symbolic systems → human–AI workflows → adaptive agents → verified software.

MOTIVATION

A failure that feels almost human

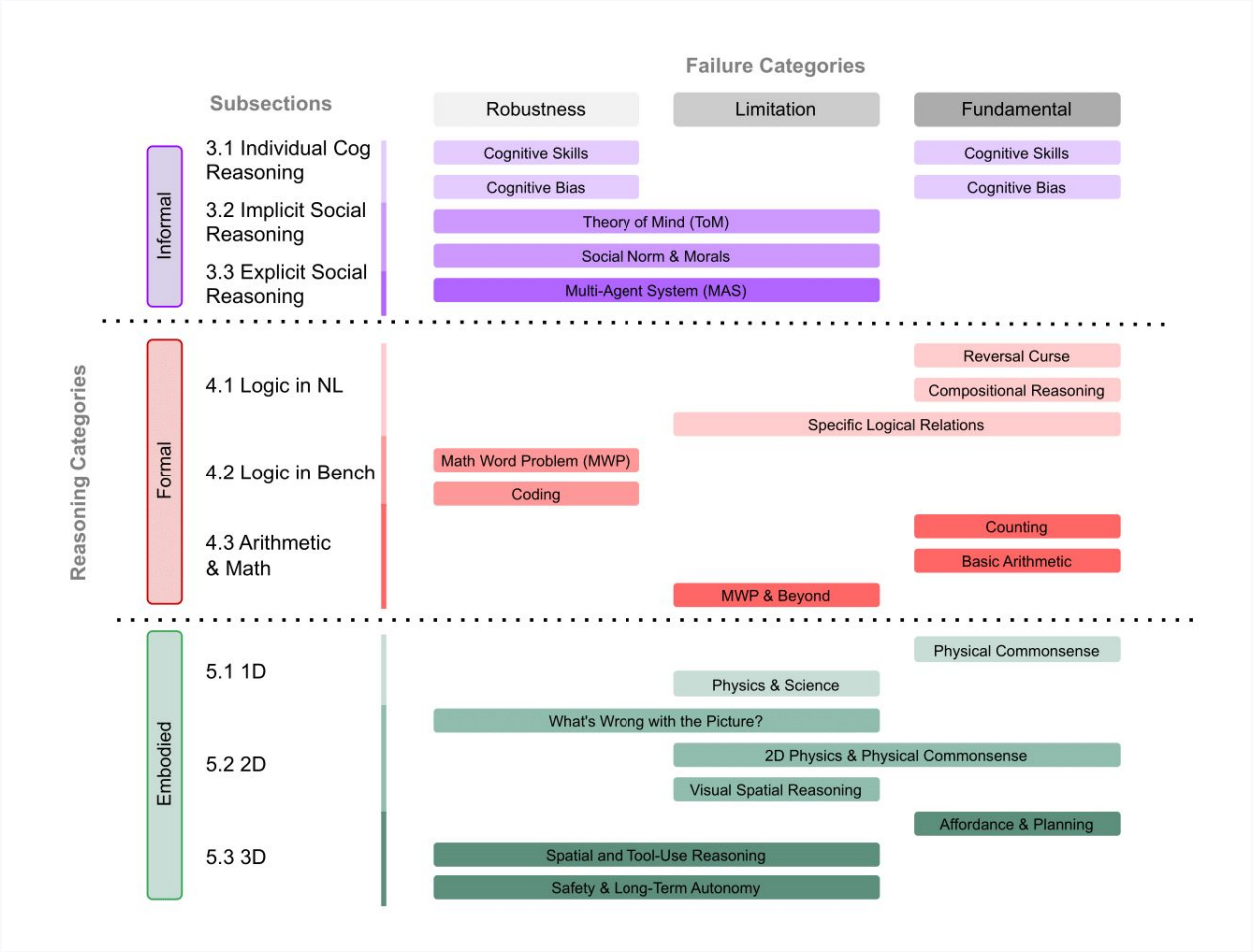
A-Not-B errors: a model keeps following an old, trivial pattern after the context changes.



MOTIVATION

Reasoning failures are not isolated bugs

They recur across intuitive, logical, mathematical, and embodied settings.



How to fix/alleviate this?

If natural-language reasoning is hard to judge reliably, can we build settings where correctness is machine-checkable?

PROMISE

The neuro-symbolic promise

Let neural models generate; let symbolic tools verify.

Natural collaboration

LLMs are strong generators: guesses, tactics, plans, code.

Formal tools are strong judges: parse, type-check, verify.

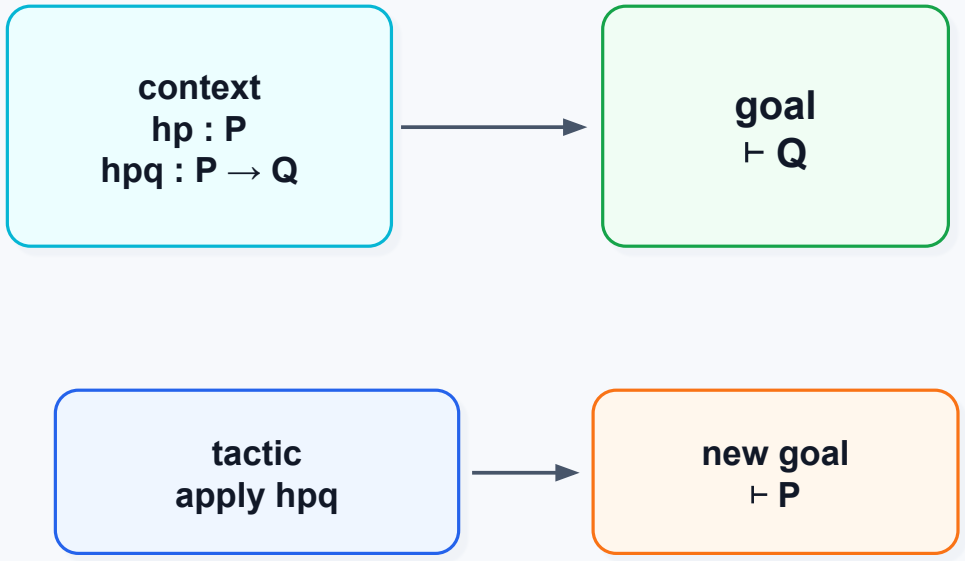
Not “neural vs. symbolic”: the point is the interface.

FORMAL TOOLS

What Lean gives the loop

A proof assistant turns reasoning into exact state transitions.

```
example (P Q : Prop)
        (hp : P)
        (hpq : P → Q) : Q := by
  apply hpq
  assumption
```



$s_{t+1} = TLean(s_t, a_t)$

The model proposes an action; Lean either accepts it and returns a new state, or rejects it with rich feedback regarding e.g. type mismatches.

FORMALIZATION

Reasoning as sequential decision making

The proof becomes a trajectory through states.

$a_t \sim \pi_\theta(a \mid s_t)$

$s_{t+1} = \text{TLean}(s_t, a_t)$

success ⇔ no goals left

ML CORE

Learning the next proof step

Classification: proof state in, tactic/action distribution out.

Training data

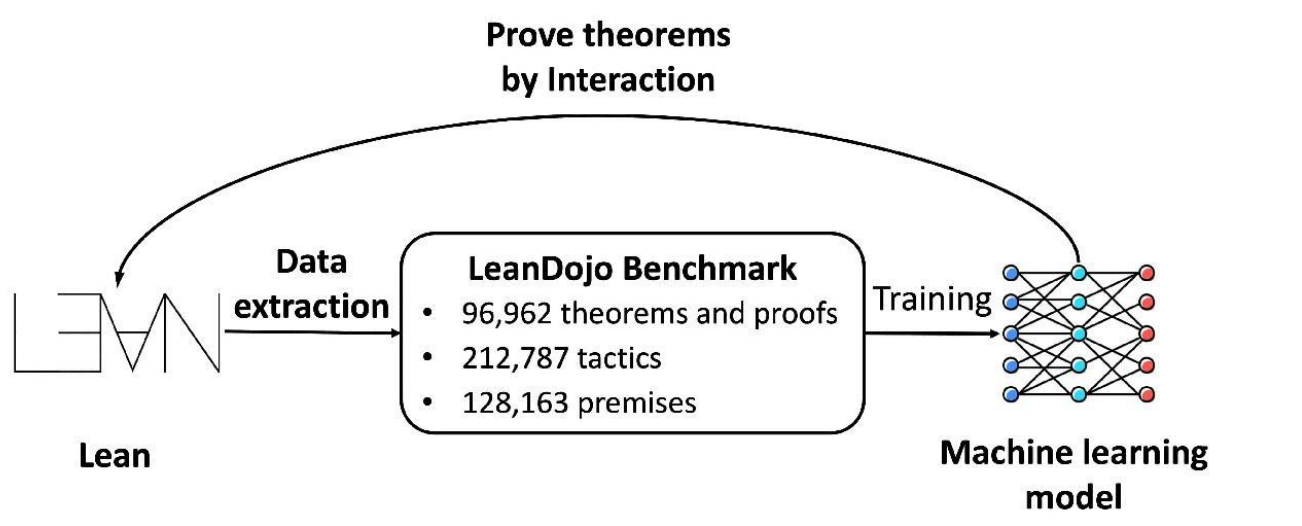
```
state: hp : P, hpq : P → Q ⊢ Q
label: apply hpq

state: hp : P ⊢ P
label: assumption
```

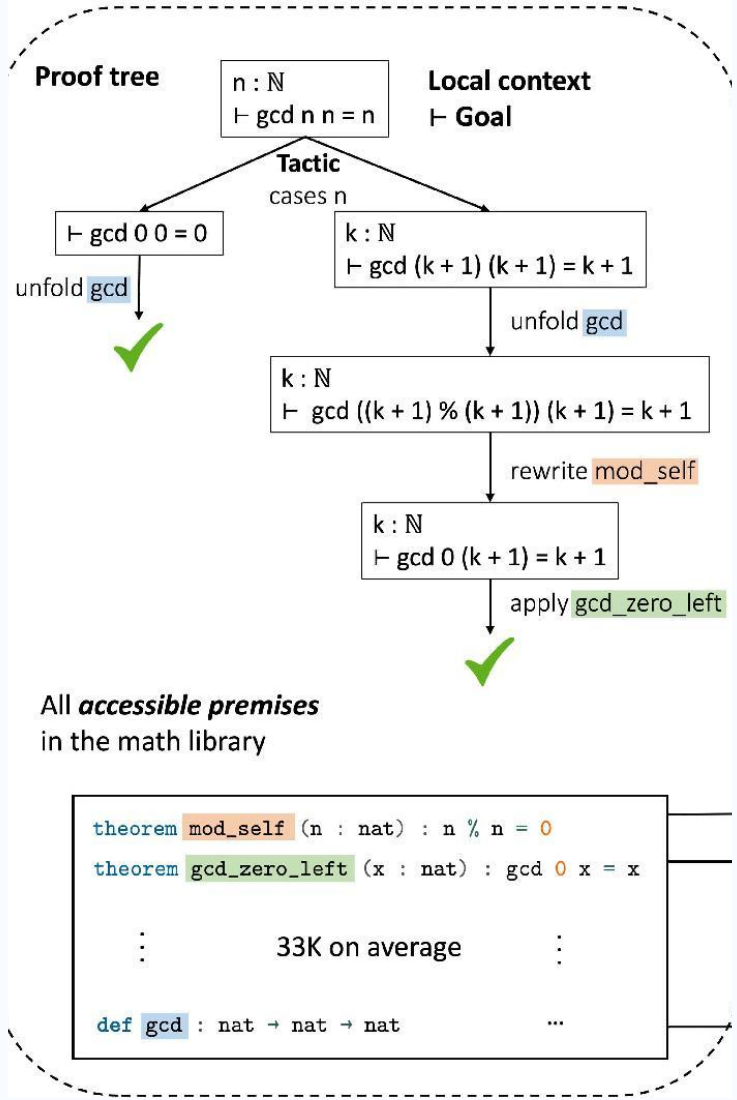
$$\pi\theta(a \mid s) = P\theta(\text{next action} = a \mid \text{proof state} = s)$$

$$L(\theta) = -\Sigma_i \log \pi\theta(a_i \mid s_i)$$

LeanDojo: opening the Lean ↔ ML loop

Infrastructure for data extraction, programmatic interaction, and reproducible theorem-proving research.

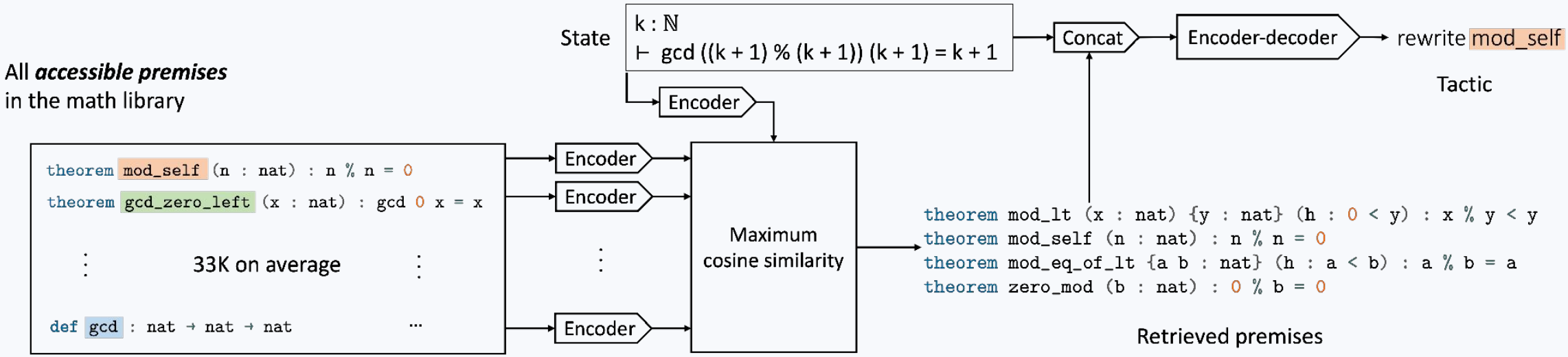




RETRIEVAL

ReProver: retrieval makes large libraries usable

A prover must choose from a vast premise library before generating a useful tactic.



CONTROL

Prediction alone is not reasoning

A legal, likely step can move the proof away from success.

A tiny theorem

$$R,\ Q_3 \to R,\ Q_2 \to Q_3,\ Q_1 \to Q_2 \vdash R$$

Trivial proof:

```
assumption
```

Tempting but bad path:

$$R \to Q_3 \to Q_2 \to Q_1$$

greedy policy chooses apply → Lean accepts the tactic

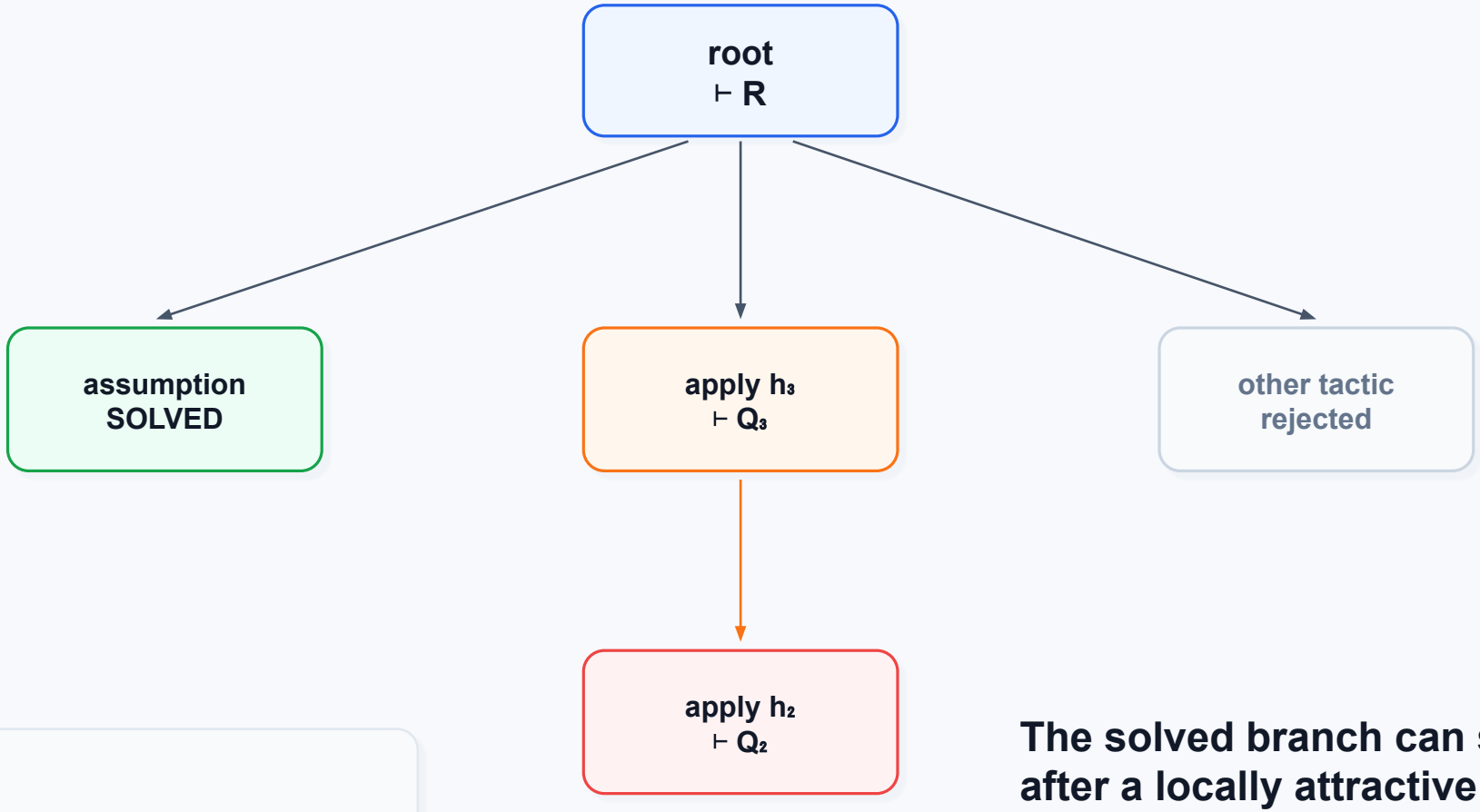
new subgoal Q_3 → eventually stuck at Q_1

Verification gives local validity; search gives strategic recovery.

SEARCH

Search + verifier: generate, check, recover

Keep alternatives alive instead of committing to the first accepted step.



Best-first score

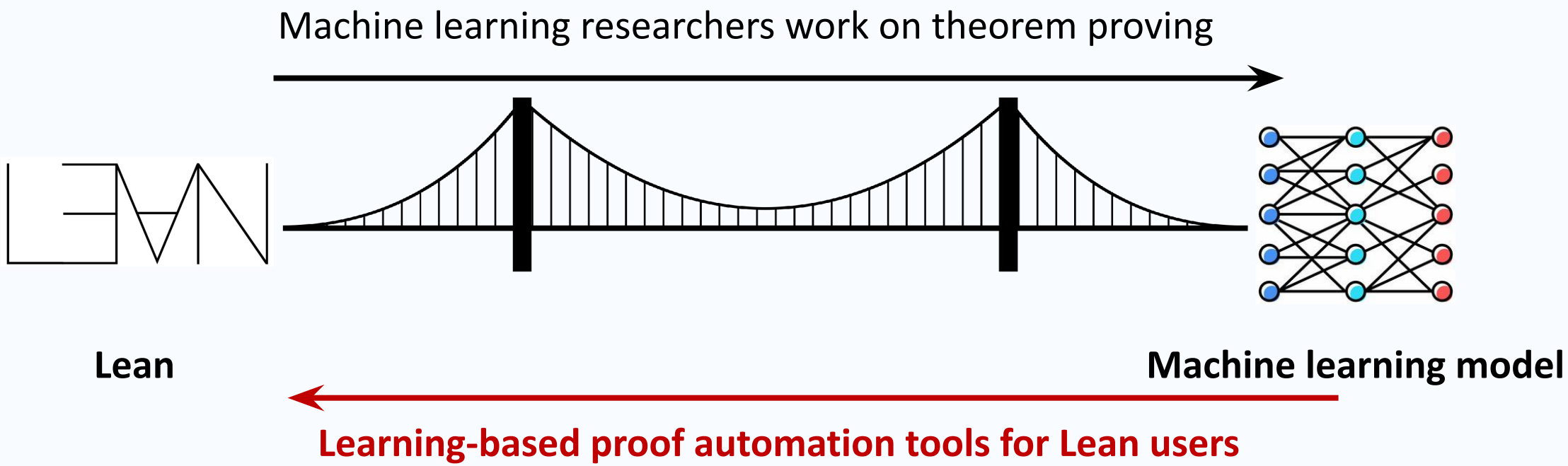
$$\Sigma \log \pi\theta(a\square \mid s\square)$$

The solved branch can still be found after a locally attractive detour.

HUMAN-AI COLLABORATION

Lean Copilot: AI inside the proof workflow

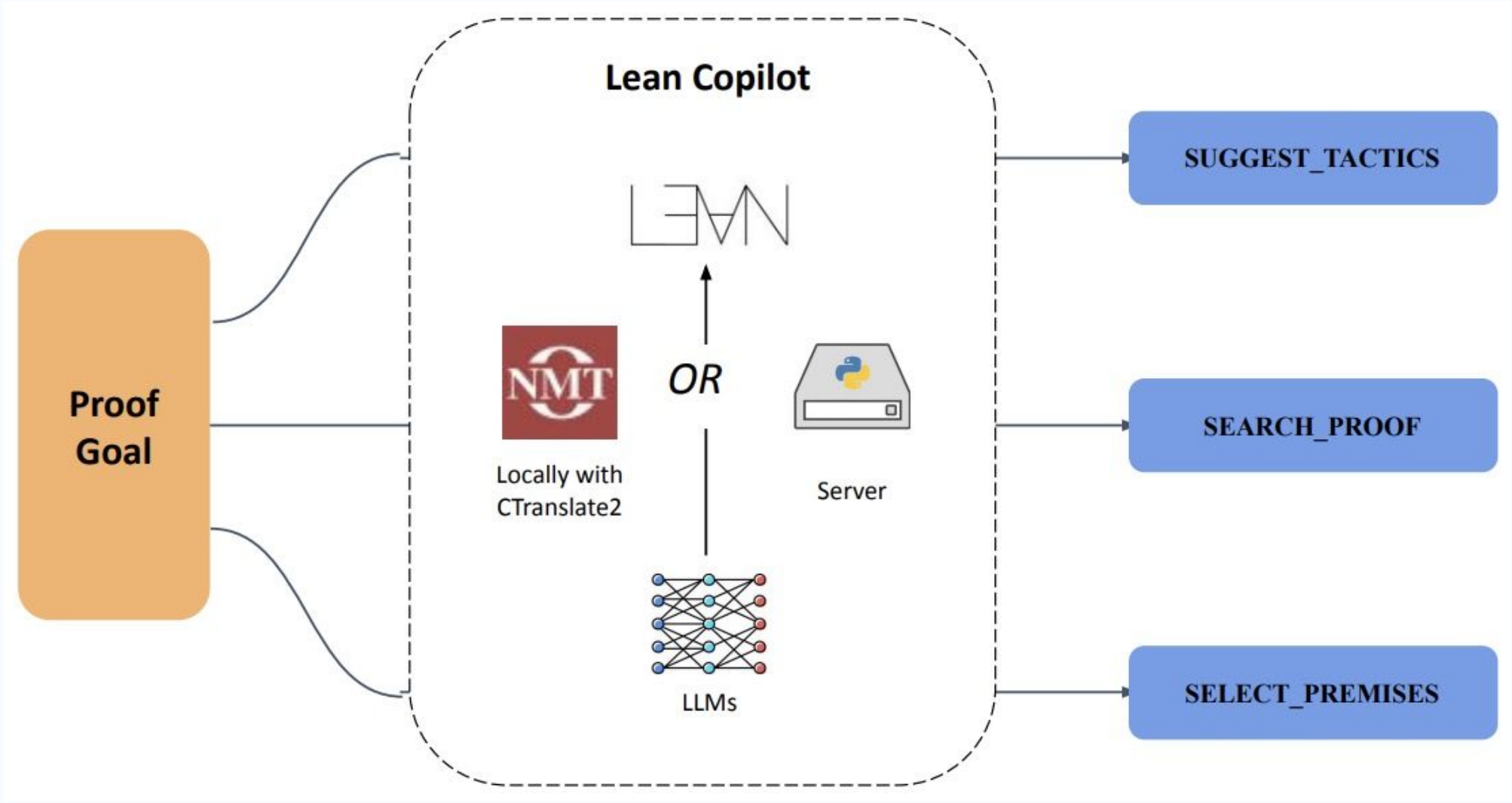
A shift from autonomous theorem proving to interactive assistance.



HUMAN-AI COLLABORATION

Lean Copilot: AI inside the proof workflow

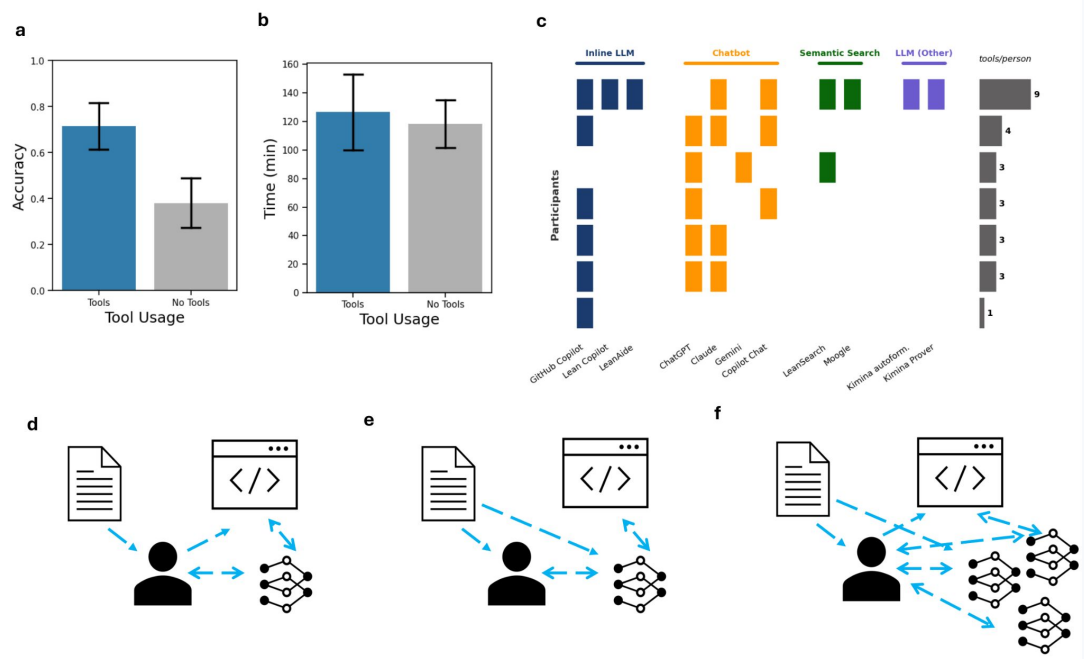
A shift from autonomous theorem proving to interactive assistance.

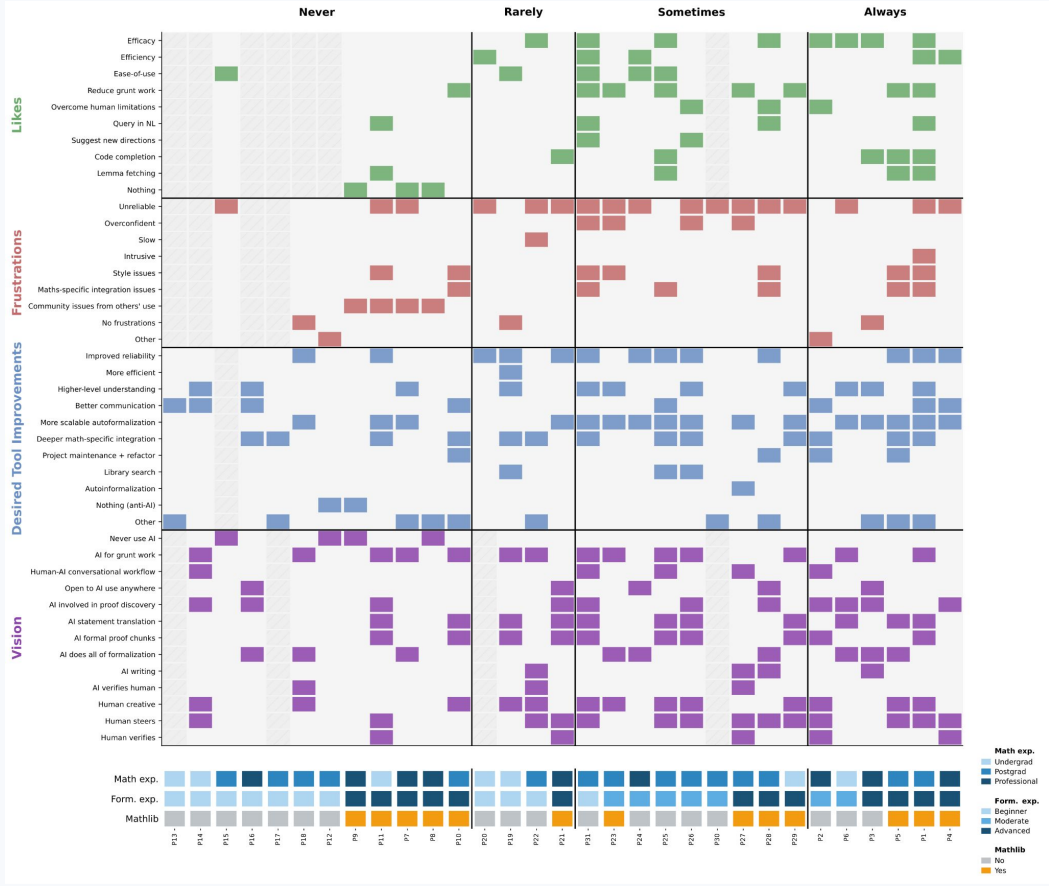


HUMAN AGENCY

Who should control the proof?

Human–AI formalization is not simply “maximize automation.”





People’s preferences are diverse, but many want AI assistance that preserves high-level control and agency over proof discovery.

PROGRESS

Progress, not just probability

A local policy needs a global sense of distance-to-goal.

Policy question

$$\pi_\theta(a \mid s)$$

What action looks likely next?

Progress question

$$\hat{d}_\varphi(s) \approx \text{steps remaining}$$

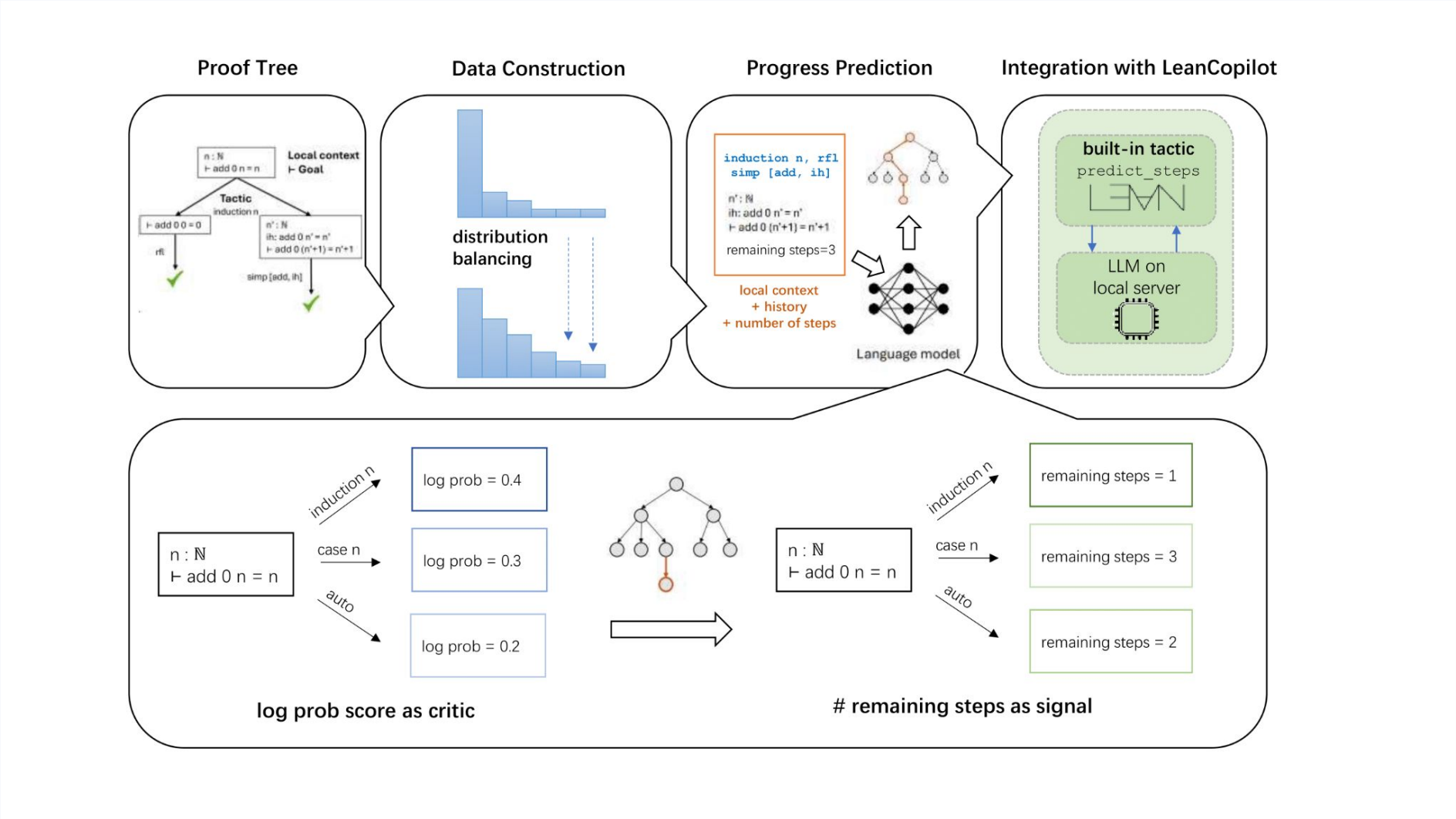
Where will this action leave me?

$$\text{priority}(s) = \Sigma \log \pi_\theta(a_i \mid s_i) - \lambda\, \hat{d}_\varphi(s)$$

PROGRESS

LeanProgress: a critic for proof search

Estimate remaining steps, then use the estimate to prioritize search.



AGENT ADAPTATION

The library keeps growing

Real mathematics is not a fixed train/test split.

Static assumption

$$D_{train} \approx D_{test}$$

A fixed model is evaluated after the world stops changing.

Lifelong setting

$$D_1 \rightarrow D_2 \rightarrow \cdots \rightarrow D_{\square}$$

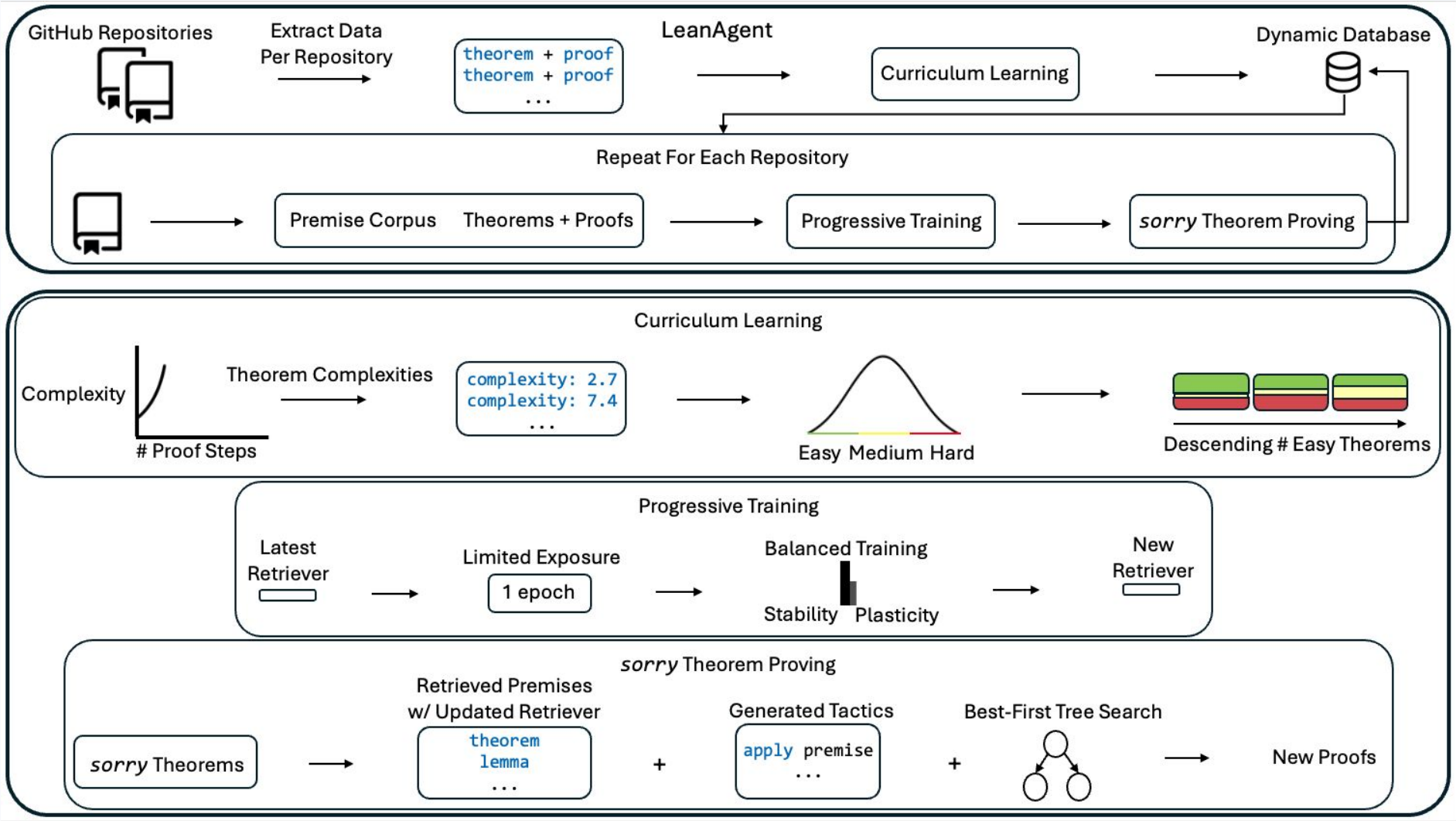
New repositories, definitions, theorems, proof styles, and dependencies keep arriving.

Challenge: learn new knowledge without forgetting old knowledge.

AGENT ADAPTATION

LeanAgent: lifelong theorem proving

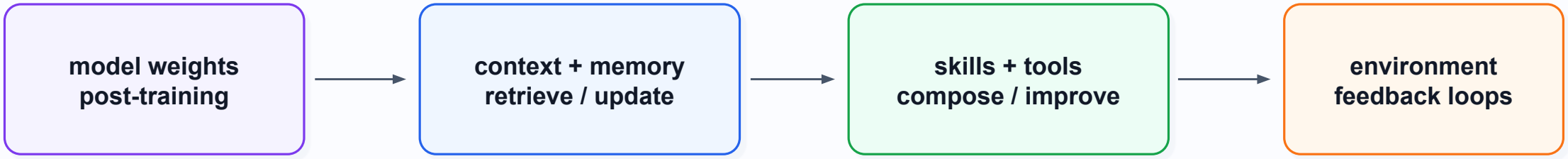
A theorem prover that adapts as mathematical knowledge expands.



AGENTS

Adaptation is a general agent problem

Where should a reasoning system change when tasks and context grow?

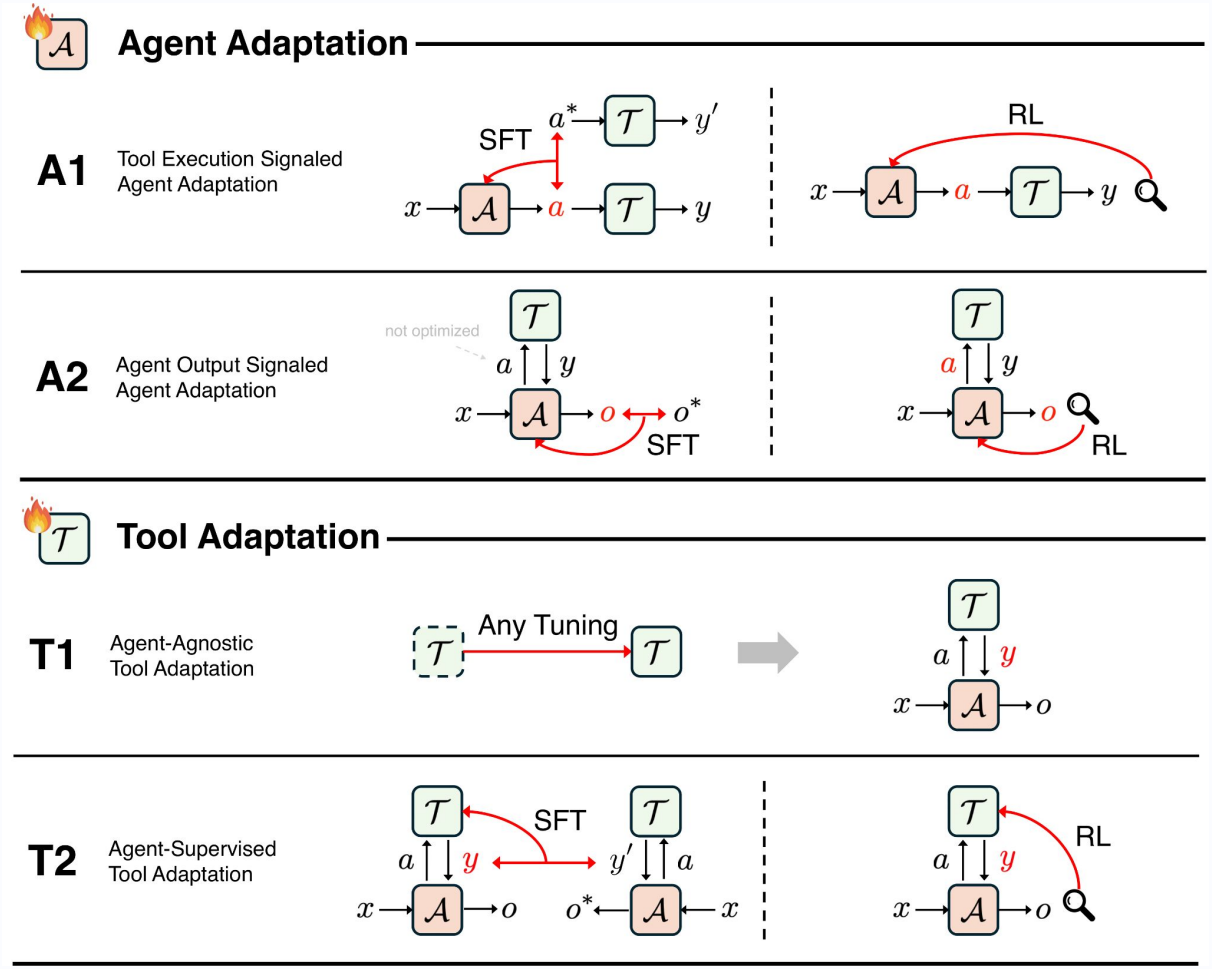


LeanProgress adapts search behavior with a progress signal; LeanAgent adapts to new libraries. The same design space appears in agentic AI broadly.

AGENTS

Adaptation is a general agent problem

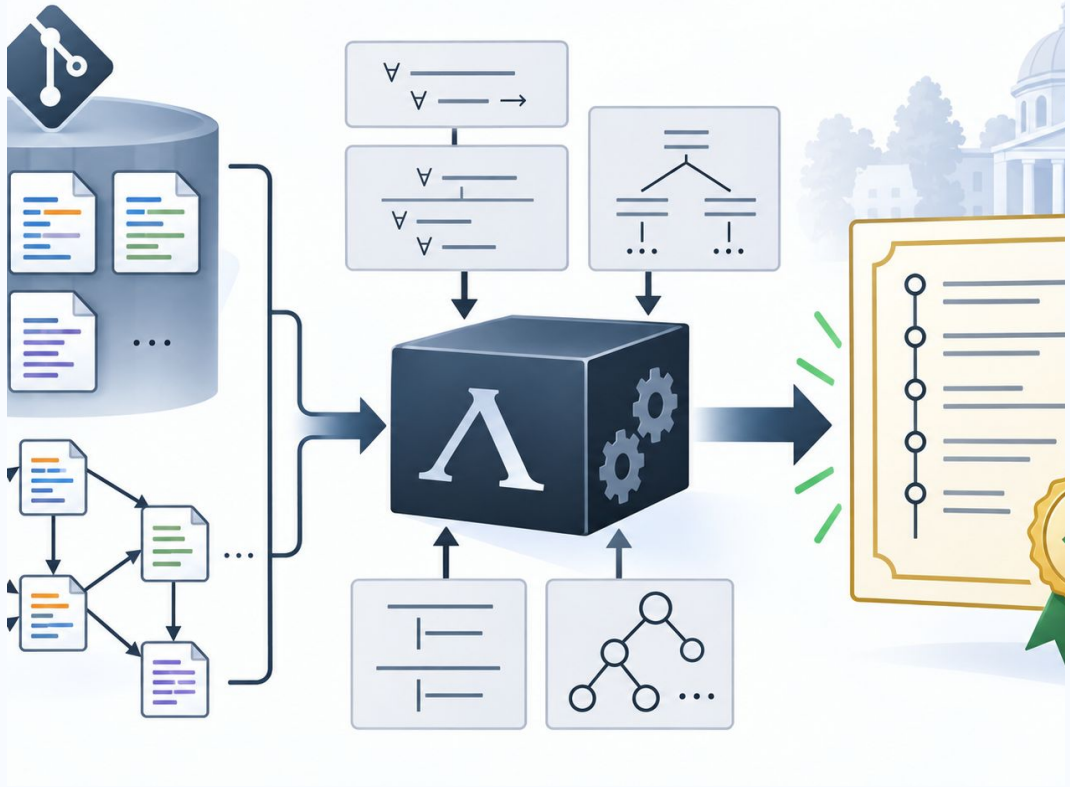
Where should a reasoning system change when tasks and context grow?

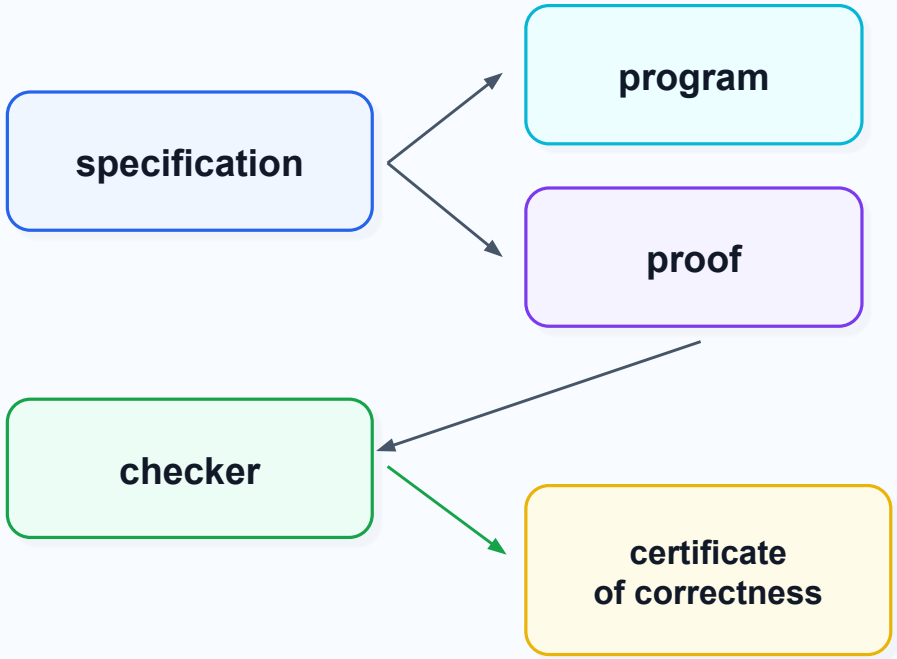


BEYOND MATH

From math proofs to verified software

The same loop applies when the artifact is code plus a proof.





Stronger guarantee: not just “the generated code passes tests,” but “the code satisfies a formal specification.”

VERIFIED CODING

Vero: repository-scale verified software

Can agents make coherent implementation-and-proof choices across modules?

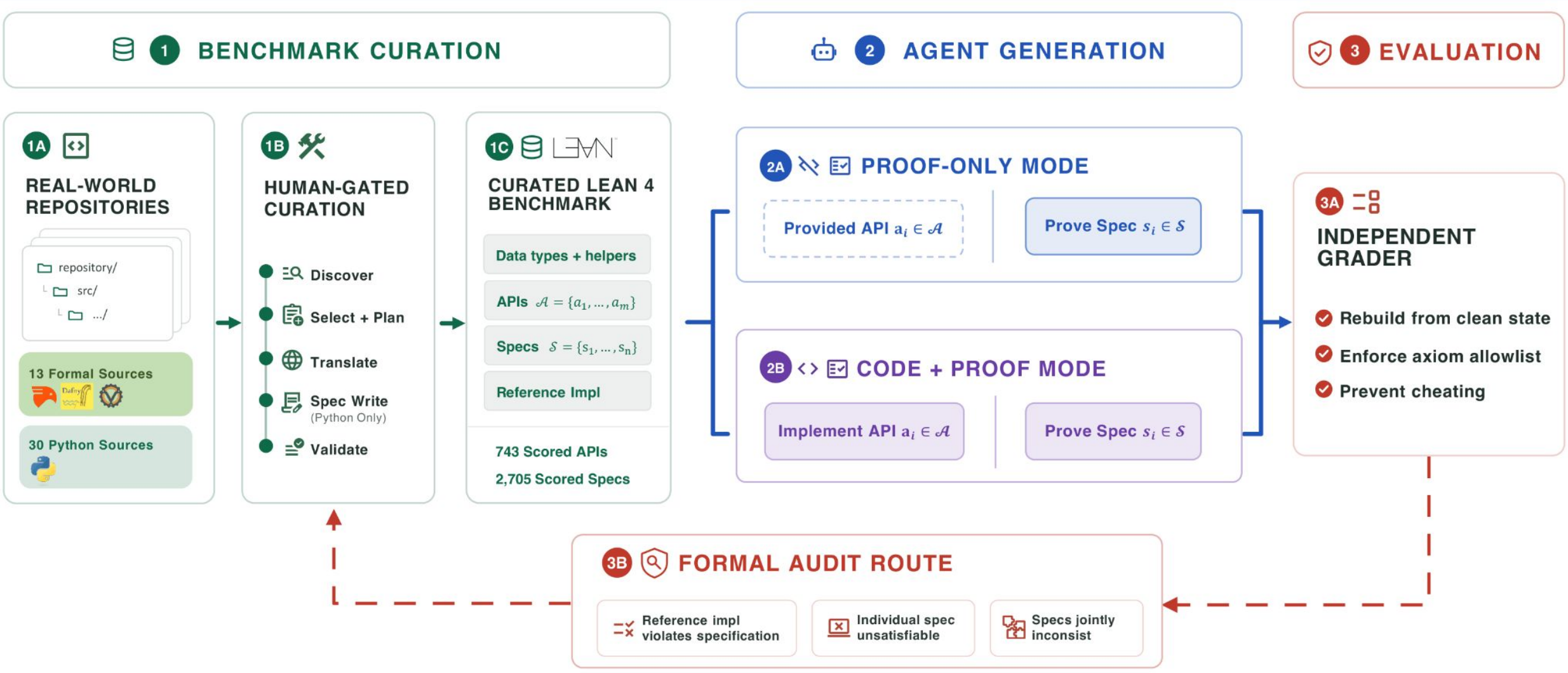
43	743	2,705	27/43
repository-level instances	APIs	formal specifications	best full solutions

Key shift: from single functions to multi-module repositories with interacting APIs, specs, and dependencies.

VERIFIED CODING

Vero: repository-scale verified software

Can agents make coherent implementation-and-proof choices across modules?

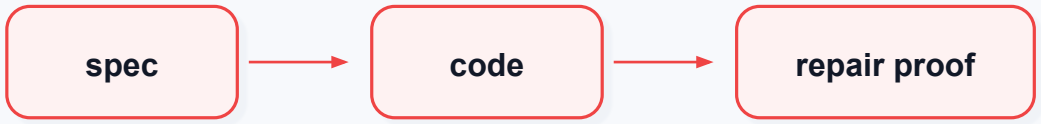


VERIFIED CODING

P³: plan program and proof together

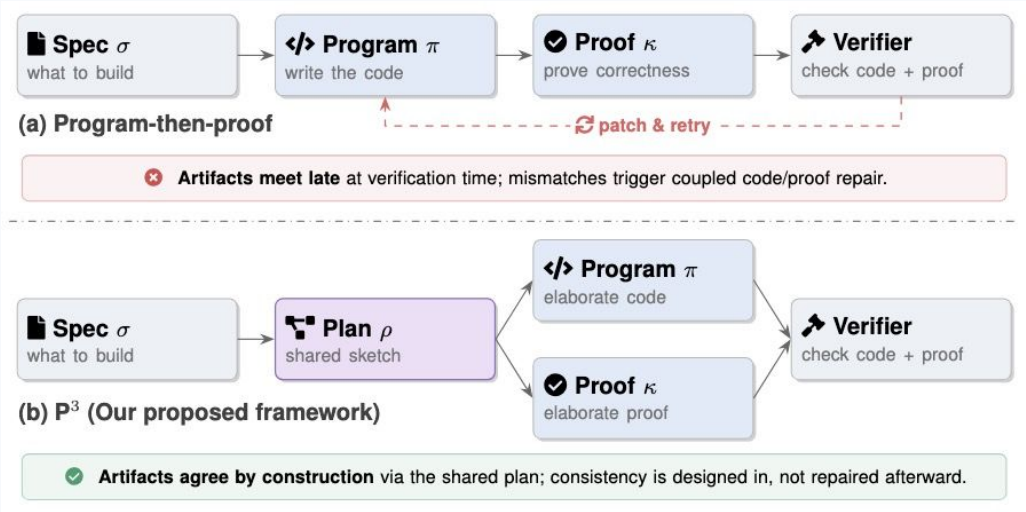
Implementation choices can make the later proof easy—or impossible.

Sequential baseline



Code-first decisions can lock in proof-hostile structure.

P³ workflow



Core lesson

Verified code generation is not two independent problems. The program and its correctness argument should be designed hand in hand.

P³ = Joint Program-and-Proof Planning

FUTURE

Where else can formal reasoning lead?

Anywhere we can specify what we want and check evidence that we got it.

Open question

Can AI explore creatively while correctness is enforced by something stronger than model confidence?

Now build the loop

Notebook walkthrough

Peiyang Song

PhD Student @ CMU LTI

psong2@andrew.cmu.edu

ECE 4424 / CS 4824 · Machine Learning
Virginia Tech · Sept. 14, 2026