INT201 Decision, Computation and Language

Lecture 13 –AI and Formal Reasoning

Dr Chunchuan Lyu

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Today [this lecture is extra, non-examinable]

- A Taste of Lean
- AI and the Future of Mathematics

Lean

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
Mathematician, UCLA

[LEAN]

Lean is helping mathematicians before the AI revolution

Lean

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
VP, Science and Strategic Initiatives, Google DeepMind

[LEAN]

Combining, LLMs and Lean, automated formal math reasoning is in the horizon

Lean

Lean is the core verification technology behind Cedar, the open-source authorization language that powers cloud services like Amazon Verified Permissions and AWS Verified Access. Our team rigorously formalizes and verifies core components of Cedar using Lean's proof assistant, and we leverage Lean's lightning-fast runtime to continuously test our production Rust code against the Lean formalization. Lean's efficiency, extensive libraries, and vibrant community enable us to develop and maintain Cedar at scale, while ensuring the key correctness and security properties that our users depend on.

Emina Torlak
Senior Principal Applied Scientist, AWS

[LEAN]

Formal verification has been used in industry

Lean

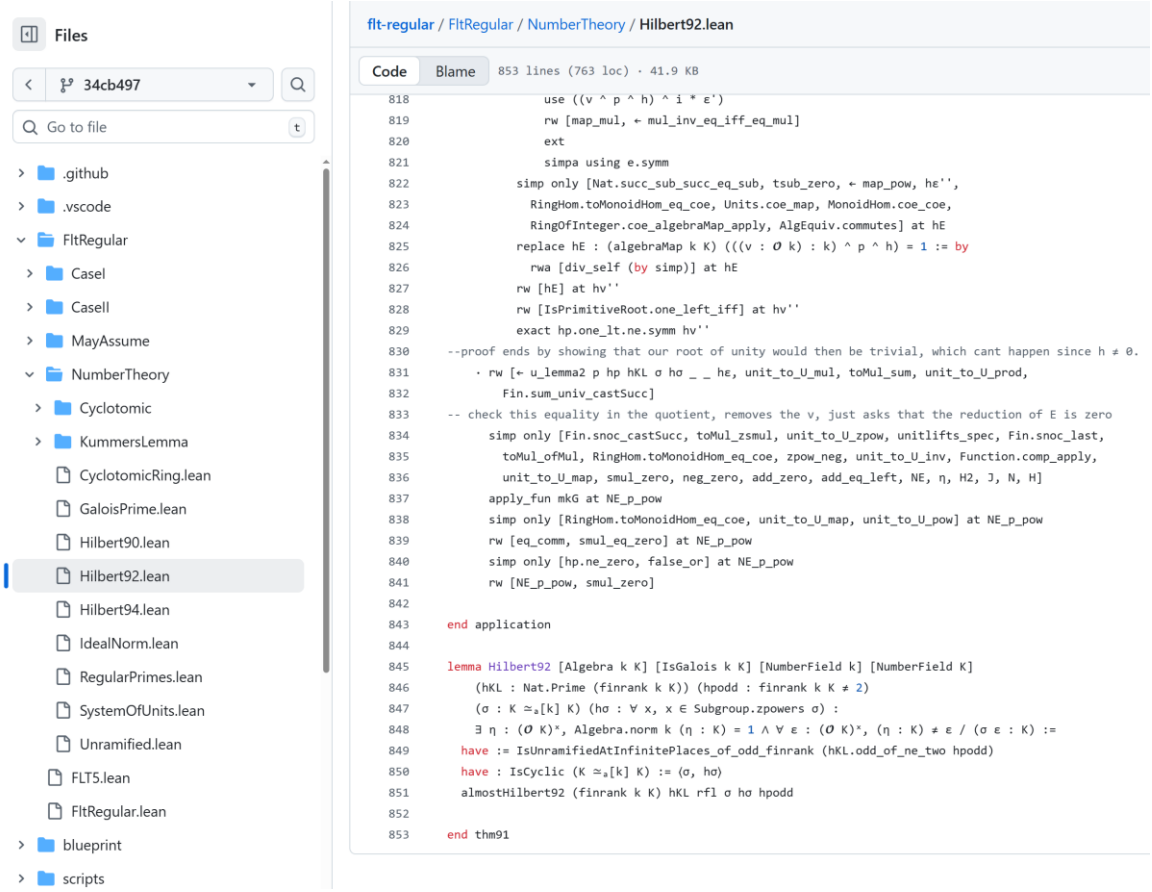
```
/-! check returns the type -/
#check 2 + 2 = 4

/-! def give name to terms of any type -/
def three := 3

/-! three is a Natural number -/
#check three

def FermatLastTheorem :=
  ∀ x y z n : Nat, n > 2 ∧ x * y * z ≠ 0 → x ^ n + y ^ n ≠ z ^ n

/-! FermatLastTheorem is a Proposition -/
#check FermatLastTheorem

/-! theorem declaries a named Proposition
  it requres a proof, but you can use sorry for place holder
-/
theorem hard : FermatLastTheorem :=
  sorry
#check hard
```

Lean

```
/-! easy can be proved
-/
theorem easy: 2 + 2 = 4 :=
  by  rfl

#check easy

/-! trivial result needs length formalization
-/
theorem and_commutative (p q : Prop) : p ∧ q → q ∧ p :=
  fun hpq : p ∧ q =>
  have hp : p := And.left hpq
  have hq : q := And.right hpq
  show q ∧ p from And.intro hq hp

#check and_commutative
```

Lean provides mechanism to check formal proofs.

Lean




```
818                 use ((v ^ p ^ h) ^ i * ε')
819                 rw [map_mul, ← mul_inv_eq_iff_eq_mul]
820                 ext
821                 simpa using e.symm
822               simp only [Nat.succ_sub_succ_eq_sub, tsub_zero, ← map_pow, hε'',
823                 RingHom.toMonoidHom_eq_coe, Units.coe_map, MonoidHom.coe_coe,
824                 RingOfInteger.coe_algebraMap_apply, AlgEquiv.commutes] at hE
825               replace hE : (algebraMap k K) (((v : 𝓞 k) : k) ^ p ^ h) = 1 := by
826                 rwa [div_self (by simp)] at hE
827               rw [hE] at hv''
828               rw [IsPrimitiveRoot.one_left_iff] at hv''
829               exact hp.one_lt.ne.symm hv''
830     --proof ends by showing that our root of unity would then be trivial, which cant happen since h ≠ 0.
831         · rw [← u_lemma2 p hp hKL σ hσ _ _ hε, unit_to_U_mul, toMul_sum, unit_to_U_prod,
832             Fin.sum_univ_castSucc]
833     -- check this equality in the quotient, removes the v, just asks that the reduction of E is zero
834           simp only [Fin.snoc_castSucc, toMul_zsmul, unit_to_U_zpow, unitlifts_spec, Fin.snoc_last,
835             toMul_ofMul, RingHom.toMonoidHom_eq_coe, zpow_neg, unit_to_U_inv, Function.comp_apply,
836             unit_to_U_map, smul_zero, neg_zero, add_zero, add_eq_left, NE, η, H2, J, N, H]
837           apply_fun mkG at NE_p_pow
838           simp only [RingHom.toMonoidHom_eq_coe, unit_to_U_map, unit_to_U_pow] at NE_p_pow
839           rw [eq_comm, smul_eq_zero] at NE_p_pow
840           simp only [hp.ne_zero, false_or] at NE_p_pow
841           rw [NE_p_pow, smul_zero]
842
843     end application
844
845     lemma Hilbert92 [Algebra k K] [IsGalois k K] [NumberField k] [NumberField K]
846         (hKL : Nat.Prime (finrank k K)) (hpodd : finrank k K ≠ 2)
847         (σ : K ≃ₐ[k] K) (hσ : ∀ x, x ∈ Subgroup.zpowers σ) :
848         ∃ η : (𝓞 K)ˣ, Algebra.norm k (η : K) = 1 ∧ ∀ ε : (𝓞 K)ˣ, (η : K) ≠ ε / (σ ε : K) :=
849       have := IsUnramifiedAtInfinitePlaces_of_odd_finrank (hKL.odd_of_ne_two hpodd)
850       have : IsCyclic (K ≃ₐ[k] K) := ⟨σ, hσ⟩
851       almostHilbert92 (finrank k K) hKL rfl σ hσ hpodd
852
853     end thm91
```

But the way, now you see why I said writing proof is like writing code ☺

Lean is not easy. Formalization takes more time from mathematician.

AI and the Future of Mathematics

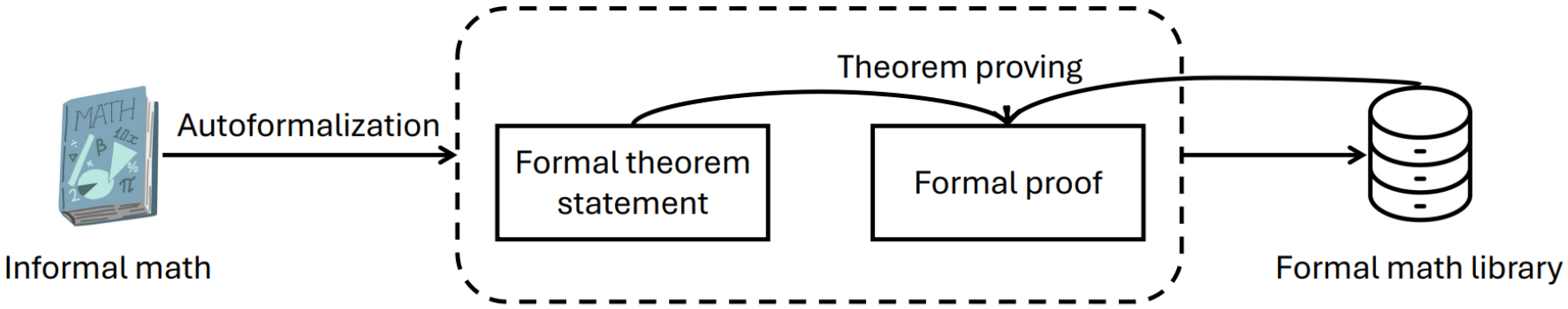


Figure 3: Common tasks using AI for formal mathematical reasoning in proof assistants.

[FMR]

AI can help with formalization and coming up with the proofs.

AI and the Future of Mathematics

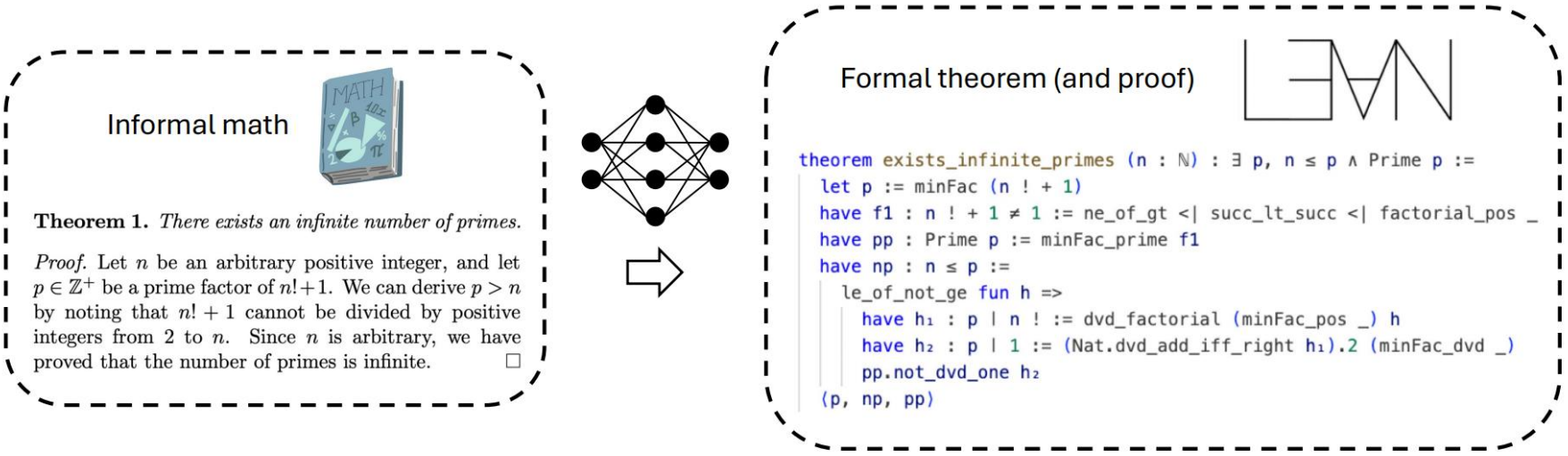


Figure 5: Autoformalization translates informal math to formal theorems and/or proofs automatically.

[FMR]

As long as the system is probabilistic (including human mathematician), there is always the possibility of making an formalization error.

However, checking the correspondence between theorem statement is sufficient.

AI and the Future of Mathematics

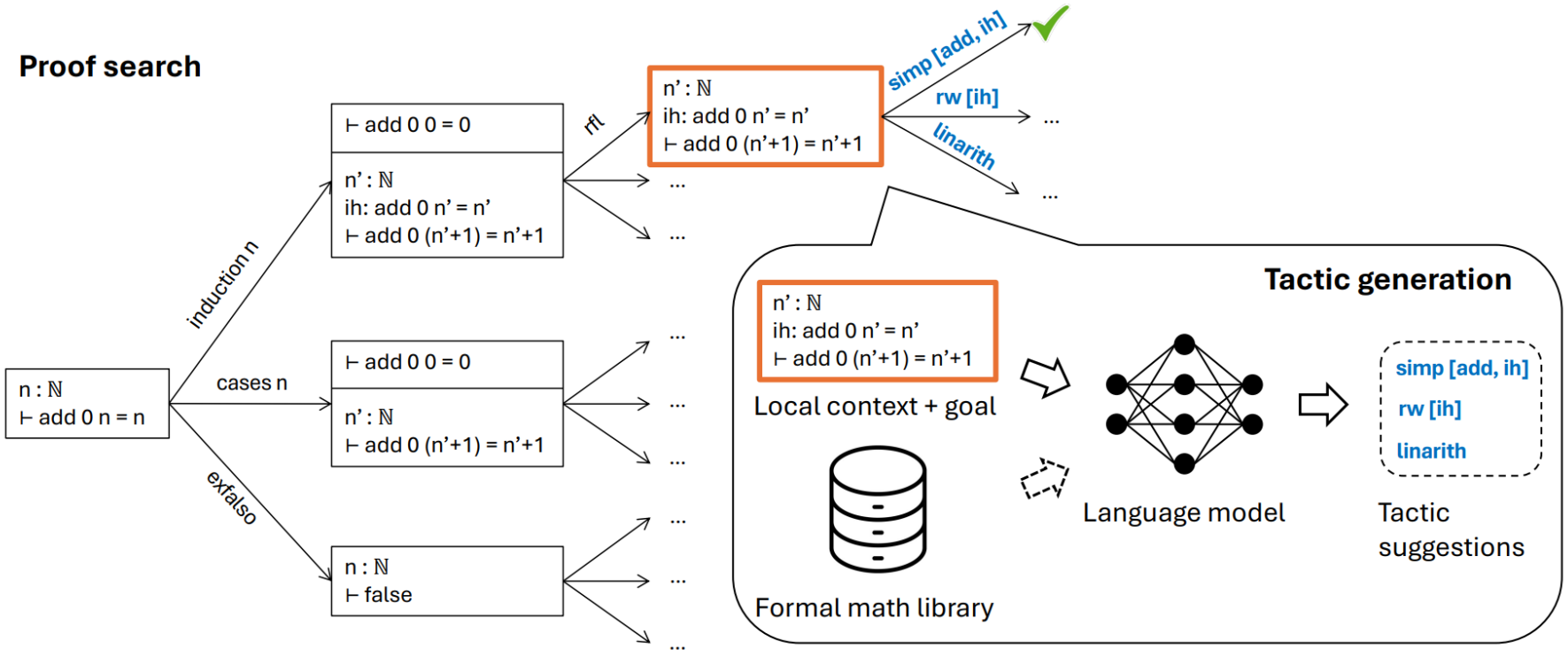
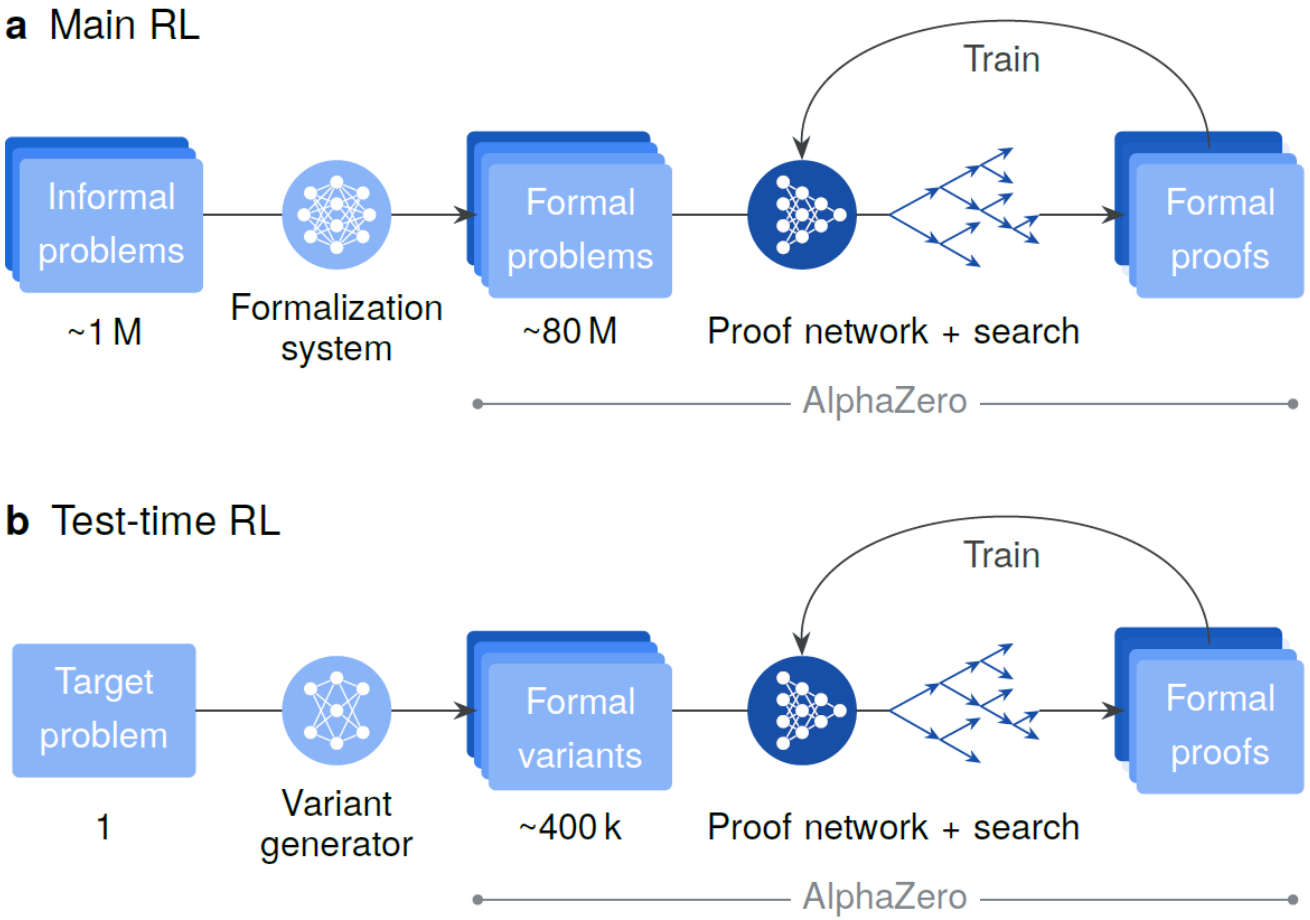


Figure 4: A neural theorem prover that combines tactic generation and proof search. This architecture is adopted by the majority of existing methods, with only a handful of exceptions [39, 40]. [FMR]

Proving is searching for a proof! Same applies to human, you don't see a proof in the first sight, you try until you find it.

AI and the Future of Mathematics



AlphaProof , 2024

Olympiad-level

You can use data to train the model, as Lean provides verification/reward on the fly, testing time learning is also possible.

AI and the Future of Mathematics

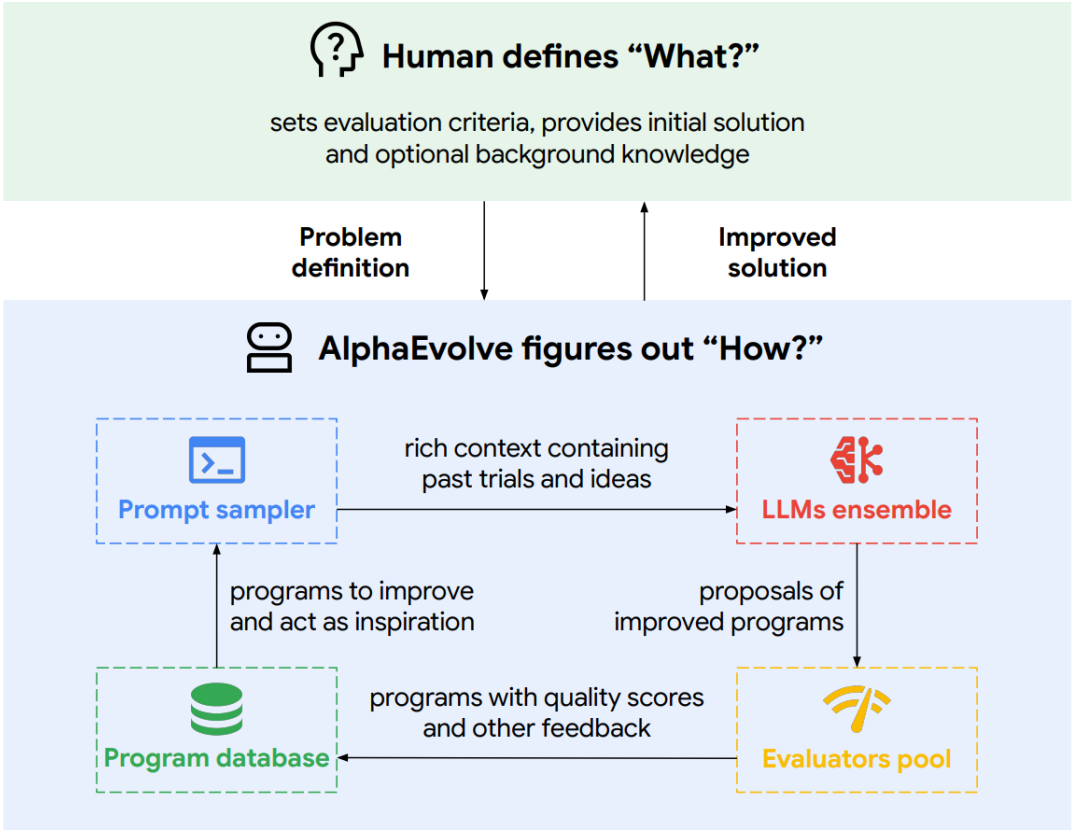


Figure 1 | *AlphaEvolve* high-level overview.

AlphaEvolve, 2025

Agentic System for Code&Math

AI and the Future of Mathematics

[MEDS]

AlphaEvolve, Nov 2025

New results for mathematicians

"These results demonstrate that large language model-guided evolutionary search can autonomously discover mathematical constructions that complement human intuition, at times matching or even improving the best known results, highlighting the potential for significant new ways of interaction between mathematicians and AI systems. "

Summary

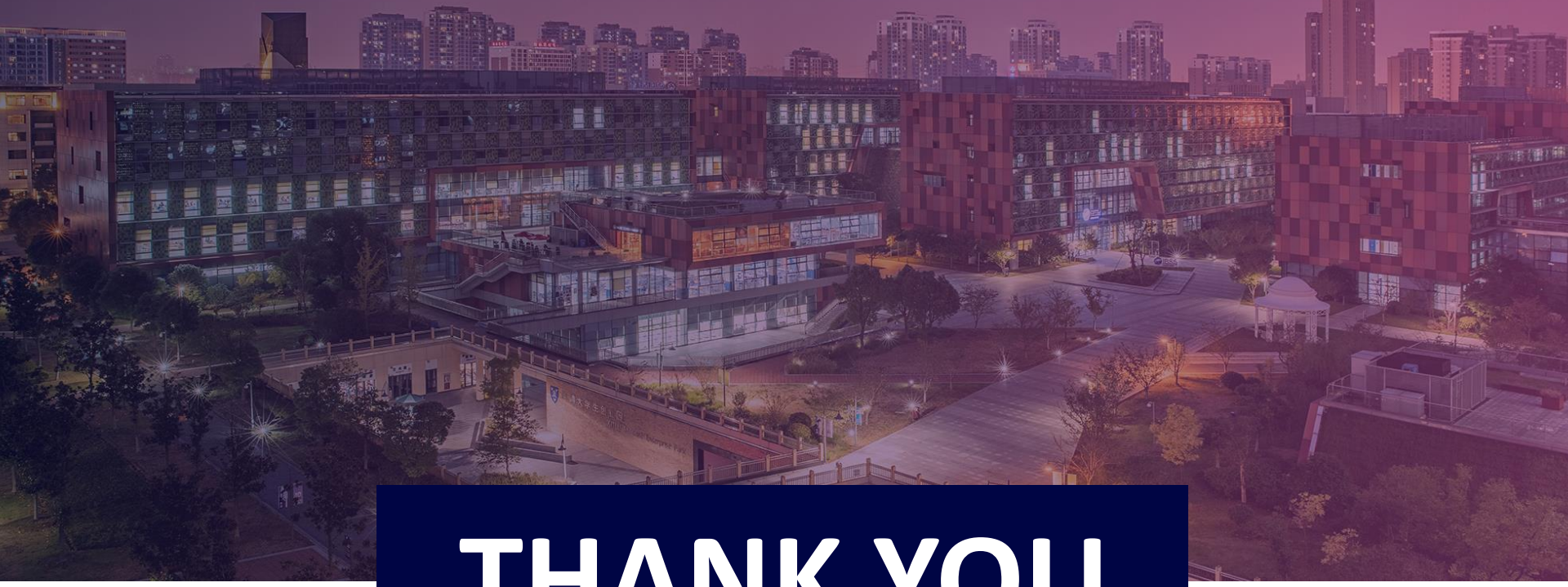
Table 1: Theorem proving: capability levels and benchmarks for evaluation.

Level	Capability	Evaluation and benchmarks
0	Checking formal proofs	Achieved by modern proof assistants
1	Assisting humans to develop proofs by suggesting definitions, lemmas, proof steps, etc.	Human-centered evaluation
2	Human-implemented tactics for automating domain-specific proof goals	Domain-specific benchmarks
3	Proving simple theorems automatically in a domain-general fashion	CoqGym [33], LeanDojo [35], MiniF2F [208], PutnamBench [209], TPTP [238], FIMO [239]
4	Contributing to formalization projects autonomously	New benchmarks of code and metadata from GitHub, similar to SWE-bench [240]
5	Solving problems and discovering new math beyond the human level	New benchmarks and evaluation methodology for unknown territories

[FMR]

Reference

- [FMR] Formal Mathematical Reasoning: A New Frontier in AI
- [AlphaProof] Olympiad-level formal mathematical reasoning with reinforcement learning
- [AlphaEvolve] AlphaEvolve: A coding agent for scientific and algorithmic discovery
- [MEDS] MATHEMATICAL EXPLORATION AND DISCOVERY AT SCALE
- [LEAN] https://lean-lang.org/



THANK YOU

Xi'an Jiaotong-Liverpool University
西交利物浦大学

XJTLU | SCHOOL OF FILM AND TV ARTS