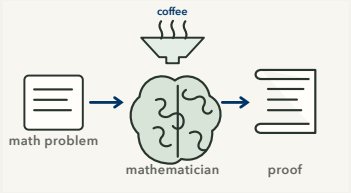
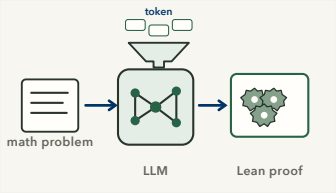
Ripple: An Open, AI-Formalized Lean 4 Framework for Computing with CRNs

Ho-Lin Chen
National Taiwan University
holinchen@ntu.edu.tw

Xiang Huang
University of Illinois Springfield
xhuan5@uis.edu xianghuang.org

DNA 32 — 32nd International Conference on DNA Computing and Molecular Programming
August 3–7, 2026 · Fayetteville, Arkansas
Technical report: arXiv:2607.13531





Being a theoretical computer scientist is an awesome job.

Being a theoretical computer scientist is an awesome job.

- They hire us to prove theorems. They don't tell us which theorem to prove.

Being a theoretical computer scientist is an awesome job.

- They hire us to prove theorems. They don't tell us which theorem to prove.
- "A mathematician is a machine for turning coffee into theorems." — Alfréd Rényi

But there is a caveat.

Your proof may haunt you—maybe for life.

Your proof may haunt you—maybe for life.

- a typo

Your proof may haunt you—maybe for life.

- a typo
- a hand-waved step

Your proof may haunt you—maybe for life.

- a typo
- a hand-waved step
- a false theorem you cite

Your proof may haunt you—maybe for life.

- a typo
- a hand-waved step
- a false theorem you cite
- a hidden assumption

Your proof may haunt you—maybe for life.

- a typo
- a hand-waved step
- a false theorem you cite
- a hidden assumption
- a hidden error

Your proof may haunt you—maybe for life.

- a typo
- a hand-waved step
- a false theorem you cite
- a hidden assumption
- a hidden error
- ...

How do we stop a proof from haunting us?

Formal proofs.

Formal means machine-checked.

What is a proof?

What is a proof?

LEFT DISTRIBUTIVITY

$\forall\, a, b, c \in \mathbb{N}, \quad a(b + c) = ab + ac$

Fix a and b. Prove the statement by induction on c.

BASE CASE c = 0

$a(b + 0) = ab = ab + a \cdot 0$ recursive definitions

INDUCTIVE STEP

Assume $a(b + c) = ab + ac$ induction hypothesis

$$
\begin{aligned}
a(b + S(c)) &= aS(b + c) && \text{addition} \\
&= a(b + c) + a && \text{multiplication} \\
&= (ab + ac) + a && \text{induction hypothesis} \\
&= ab + (ac + a) = ab + aS(c) && \text{associativity · multiplication}
\end{aligned}
$$

$\therefore \quad a(b + c) = ab + ac$

What is a proof?

LEFT DISTRIBUTIVITY

$\forall\, a, b, c \in \mathbb{N}, \quad a(b + c) = ab + ac$

Fix a and b. Prove the statement by induction on c.

BASE CASE c = 0

$a(b + 0) = ab = ab + a\cdot 0$ recursive definitions

INDUCTIVE STEP

Assume $a(b + c) = ab + ac$ induction hypothesis

$$
\begin{aligned}
a(b + S(c)) &= aS(b + c) && \text{addition}\\
&= a(b + c) + a && \text{multiplication}\\
&= (ab + ac) + a && \text{induction hypothesis}\\
&= ab + (ac + a) = ab + aS(c) && \text{associativity · multiplication}
\end{aligned}
$$

$\therefore \quad a(b + c) = ab + ac$

boring · mechanical

(virtues of a proof)

A healthy mix of English and math

=

informal proof

found in **most** books and papers

What is a formal proof?





Lean



Lean

Lean is a strict logical system: every step must mesh, with no gaps left unchecked.

What do we do with Lean?

What do we do with Lean?

- Write proofs in Lean.

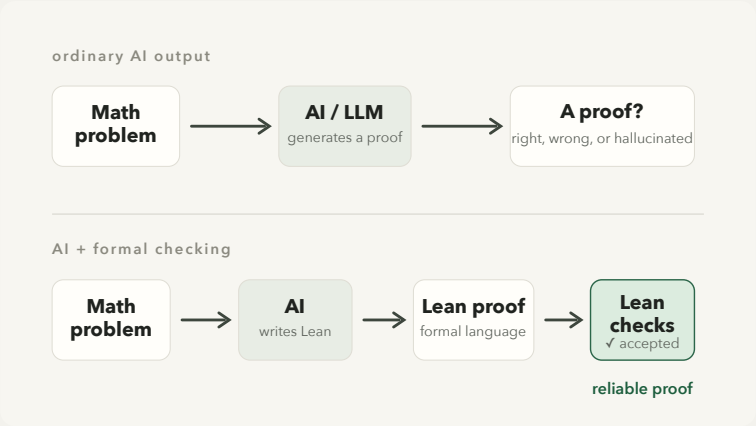
What do we do with Lean?

- Write proofs in Lean.
- boring · mechanical (same old, same old)

What do we do with Lean?

- Write proofs in Lean.
- boring · mechanical (same old, same old)
- **This time, you can do it with AI!**

AI writes. Lean checks.



But a proof checker is not yet a domain library.

But a proof checker is not yet a domain library.

Mathlib, Lean's mathematical library, does not yet cover every domain—for example, chemical reaction networks or population protocols.

The missing layer

Definitions and reusable lemmas for proving properties of CRNs and population protocols.



We treated it as an experiment.



What did we formalize?

CRN-computable numbers

SOURCE	FORMALIZED RESULT
Huang, Klinge, Lathrop, Li, Lutz (2017)	CRN-computable reals
Huang, Klinge, Lathrop (2019)	CRN = analog computer
Huang & Huls (2022)	Large-population protocols
Huang & Migunov (2024)	Analog-to-protocol compiler
Haisler, Huang, Migunov et al. (2025)	Selective dual-railing
Chen & Huang (2026)	Bounded analog complexity
QBee compilation pipeline	Automated quadraticization
Zero-init no-collapse (new)	Bounded CRN non-degeneracy
Albenque & Gerin (2012)	Urn model / algebraic numbers
Cassels (1960)	Catalan equation descent
ζ(3), π, e, ln 2, γ, Catalan, Dottie	Computable constants

Population-protocol majority

SOURCE / RESULT
Angluin, Aspnes, Eisenstat (2008) Approximate majority
Doty et al. (FOCS 2021) Exact majority
Burman, Chen, Doty et al. (2021) Self-stabilizing leader election
Kanaya et al. (SSS 2025) Self-stabilizing majority

Turing universality

Bournez, Graça, Pouly (2017)	ODE universality
Soloveichik et al. (2008)	Stochastic CRN universality

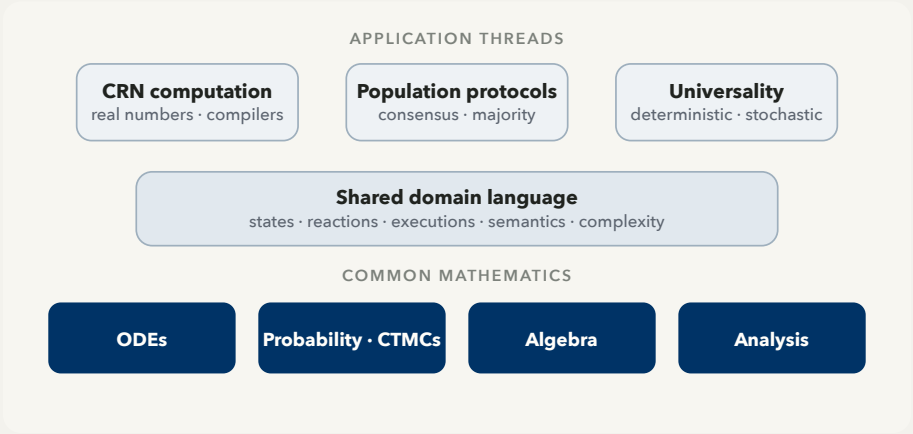
Kurtz's mean-field theorem

Kurtz (1970)	Weak convergence
Kurtz (1978)	Strong approximation
CLT scale	Mean-square bound

Supporting mathematics

Frobenius · modular forms · CTMC martingales

Find the shared mathematics.



What grew inside Ripple?

≈ 920	≈ 768K	≈ 23.4K
Lean files	lines of Lean	top-level declarations

CURRENT RIPPLE SCALE · EXPERIMENTAL AND STILL GROWING

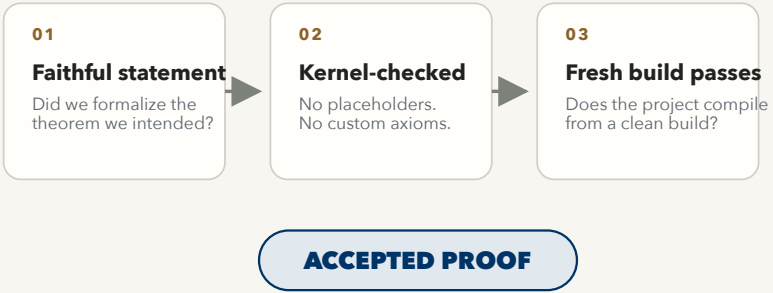
What grew inside Ripple?

≈ 920	~~≈ 768K~~ 1M+	≈ 23.4K
Lean files	lines of Lean	top-level declarations

CURRENT RIPPLE SCALE · EXPERIMENTAL AND STILL GROWING

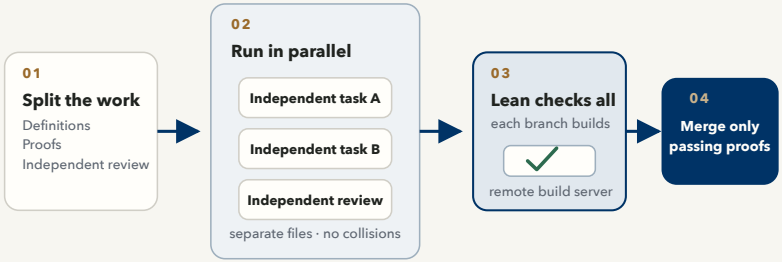
So, what does our workflow look like?

1. What counts as an acceptable proof?

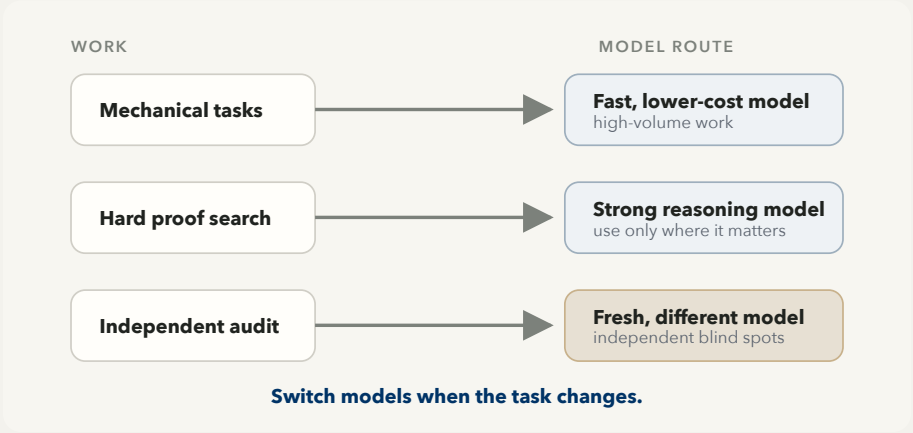


2. How do we finish checked proofs sooner?

Parallel work helps only when tasks stay independent—and every result still passes Lean.



3. Cost is a routing problem.



AUGUST 3, 2026 · ONGOING DISCUSSION

One of OpenAI's ten announced advances is now being disputed.



OpenAI announced ten results on August 1. A new critique appeared today. The community has not yet reached a verdict.

We do not yet know where the issue lies.



One possible failure mode is a valid proof of the formal statement—but a formal statement that does not faithfully capture the original problem.

Lean checks the proof. Domain experts must check the target.

Sources: OpenAI, August 1, 2026 · critique posted August 3, 2026

Future work: keep humans in the loop.

The live dispute turns "human review" from a general precaution into a concrete research requirement.

Human review	Domain experts verify that the formal statement faithfully captures the intended theorem.
Refactor	Consolidate duplicated infrastructure.
Curate	Distill essential structures into a clean, minimal library.
Track Mathlib	Continuously test Ripple against upstream updates.
Upstream	Contribute CTMC, Kurtz, and Frobenius foundations to Mathlib.

Kernel checking is necessary. Domain review is necessary too.

Thank you for your time.

AI writes. Lean checks. Ripple gears up the domain.

WE GRATEFULLY ACKNOWLEDGE OUR FUNDING AGENCIES

NSTC 國家科學及技術委員會
National Science and Technology Council

NSTC (Taiwan) · Grant 113-2221-E-002-204-MY3

U.S. DEPARTMENT OF ENERGY | Office of Science

U.S. DOE Office of Science, ASCR
Grant DE-SC0024278