Multi-Agent Theorem Proving in Lean 4

Riyaz Ahuja, Shivansh Gour

17-599 Spring 2026

The Prime Number Theorem

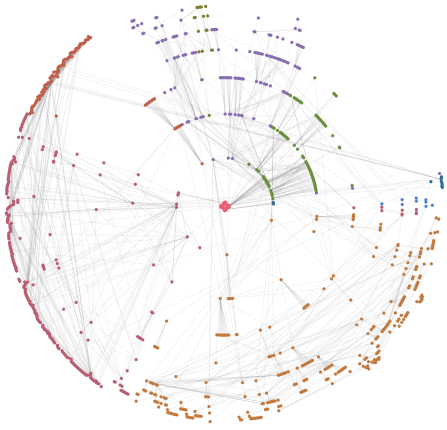
Prime Number Theorem

The number of primes at most n is asymptotic to $n/\ln n$.

- Proven by Hadamard & de la Vallée Poussin in 1896.
- However, was conjectured nearly a century prior: (Gauss and Legendre).
- The gap: 100 years of shared, distributed inftrastructure - Riemann's ζ-function, contour integration, entire function theory, etc.

The same pattern at scale with LLMs

Agentic LLM provers exist but treat each proof obligation as an independent search problem.



Lean dependency graph for the PNT formalization. Radial distance = depth; angular distance = namespace similarity.

Theorem Proving as an Allocation Problem

Central claim

Large-scale theorem proving is not proof search alone. It is an allocation problem over interdependent proof obligations, heterogeneous agents, private information, and individual incentives.

Questions we address:

- What mechanisms are required to coordinate self-interested agents for the social good (theorem proving)?
- Can MARL find efficient equilibria in that mechanism?
- How do these abstract mechanisms perform in real formalization efforts?

Contributions (in brief):

- Formal model (MATPG)
- Bilateral contract market and MMATPG
- PSRO + MAPPO agent for MMATPG.
- Real Lean 4 deployment & evaluation

Background: Lean 4 and Neural Theorem Proving

Lean 4 / Mathlib

- Proof assistant with dependent type theory.
- Correctness guaranteed by a minimal kernel.
- Mathlib: ≈1.9M lines, growing ≈500K/year.

Neural theorem proving

- Tactic-level provers (GPT-f, PACT, ReProver): treat each lemma as an isolated search; strong on single-lemma benchmarks (MiniF2F).
- Project-level provers (Aristotle, OpenGauss, Claude Code): read full repo context, iterate with Lean — but no known structured coordination between agents.

Mechanism design & Price of Anarchy

- Roughgarden & Tardos (2002); Koutsoupias & Papadimitriou (1999): PoA as the right lens for decentralized efficiency.
- Myerson (1981): incentive-compatible mechanisms that align private incentives with social goals.
- **Our contribution:** collateralized contracts as the mechanism; publication externality as the market failure being corrected.

Formalism: Sequents and Libraries

Core objects

- **Formula** ϕ: a statement in a formal language (e.g. Lean `Prop`). Well-defined w.r.t an involution $\neg$.
- **Sequent** $[\Gamma \vdash \phi]$: proof obligation — prove ϕ from context Γ.
- **Dependency** $\delta_t([\Gamma \vdash \phi]) \subseteq \Gamma$: what a proof relies on.

Agent action primitives

$\texttt{Prove}(s,\tau)$	Attempt sequent s for τ steps
$\texttt{Conj}(s,\tau)$	Propose a helper lemma
$\texttt{Pub}(\mathcal{C},R)$	Push private results public
$\texttt{Qry}(s)$	Estimate success prob. & time

Library state $\mathcal{L}_t = (\mathcal{C}_t, \mathcal{RS}_t, \delta_t, \Theta_t)$

- $\mathcal{C}_t$: stated / conjectured formulas.
- $\mathcal{RS}_t$: resolved (proven) sequents.
- δ_t: dependency witnesses.
- Θ_t: solve-time and credited agent.

Note that $\phi \in \mathcal{C}_t \implies \neg\phi \in \mathcal{C}_t$ and $[\Gamma \vdash \phi] \in \mathcal{RS}_t \implies [\Gamma \vdash \neg\phi] \notin \mathcal{RS}_t$. Additionally, libraries are monotone and upwards closed.

Key distinction

$\mathcal{RS}_t$ = locally proven. $\mathcal{FRS}_t$ = *fully* resolved. Publication is dependency-closed: publishing s publishes its entire proof chain public.

Centralized Baseline: MATP as a POMDP

A single planner allocates all agents.

- Makespan objective: $C_{\max} = \inf\{t : \Phi^* \subseteq \mathcal{FRS}_t^0\}$.
- Immediate publication is without loss of generality for the planner.
- **NP-hard** by reduction from $P||C_{\max}$.
- Lower bounds: work W (total processing) and serial depth D (latent revelation chains).

Reactive List Scheduling (RLS)

1. Queue $\mathcal{K}_0 = \Phi^*$.
2. Assign idle agents to highest-priority unresolved known tasks.
3. On proof completion, add revealed dependencies to $\mathcal{K}$.

RLS guarantee

$$\mathbb{E}[C_{\max}^{\mathrm{RLS}}] \leq \left(2 - \tfrac{1}{m}\right) \mathbb{E}[\mathrm{OPT}].$$

Analogous to Graham's list scheduling, with *informational* precedence replacing operational precedence.

MATPG: Decentralized Theorem Proving as a POSG

Drop the planner. Each agent acts independently:

$$G_{\mathrm{MATPG}} = (N,\ S,\ \{A^\alpha\},\ \{O^\alpha\},\ T,\ T^0,\ \{Z^\alpha\},\ \{r^\alpha\},\ \gamma)$$

- Same state/transition dynamics as MATP.
- Agent α observes $(L_t^\alpha, \Omega_t^\alpha, L_t^0)$ — own library, query results, public library. **Other agents' private state is hidden.**

Credit-based reward

$$r^\alpha_{\mathrm{credit}}(s_t, a_t) = \sum_{\substack{s \in \mathcal{FRS}^{\mathbf{0}}_{t+1} \setminus \mathcal{FRS}^{\mathbf{0}}_t \\ \Theta^{\mathbf{0}}_{t+1}(s) = (\cdot, \alpha)}} v(s).$$

Agent earns value only for public sequents *credited to it*.

Efficiency metric

$$\mathrm{PoA} = \frac{\sup_{\pi^* \in \mathsf{NE}} \mathrm{SC}(\pi^*)}{\inf_\pi \mathrm{SC}(\pi)}$$

$$\mathrm{SC}(\pi) = \mathbb{E}^\pi[C_{\mathsf{max}}].$$

The incentive problem

Credit rewards only compensate your own credited outputs. They miss the social value your publications create for others — which can lead to withholding useful intermediate results.

The Publication Externality

Illustrative instance:

$v_\alpha(s_i) = 0$ for intermediate results.
Publishing a byproduct costs effort and may help a competitor.

Theorem 3.6 (Externality exists)

If δ^* induces non-trivial cross-agent dependencies, publication externalities exist: there are states where publishing a result strictly helps others but does not help the publisher.

What happens at equilibrium

Under credit rewards, each agent may rationally work independently and withhold intermediate results. Because publication is dependency-closed, they can still eventually publish s_0 — but only after privately re-deriving the full chain. No collaboration.

Empirical result

We observe PoA of ≈ 2.91 and only 50.8% of targets resolved.

Collateralized Bilateral Contract Market

A **contract type** $\chi = (s, T, \ell)$ is a unit claim on the event $E_{s,T} = 1[s \in \mathcal{RS}_T^0]$.

Outcome	Long χ^+	Short χ^-
s is public by T	$+(1-\ell)$	$-(1-\ell)$
s is not public	$-\ell$	$+\ell$

Full collateralization: margin $B^\alpha \geq 0$ always enforced.

Market-Mediated MATPG (MMATPG)

Augment MATPG with portfolios, public order book, and balance-change reward:

$$r^\alpha_{\text{mkt}}(w_t, a_t) = B^\alpha_{t+1} - B^\alpha_t.$$

How it fixes the externality

1. β buys long exposure on s (expressing demand for its resolution).
2. α acquires long exposure, proves s, *publishes*, collects settlement.
3. Settlement requires $s \in \mathcal{RS}_T^0$ — private proof **without publication does not collect**.

Publication of intermediate results becomes privately profitable. Market price emerges endogenously from supply and demand.

Empirical result

Equilibria under the market mechanism achieve PoA ≈ 1.29 and 96.7% resolution.

MARL Framework: Approximating CCE in the MMATPG

Why not Nash? Computing NE is PPAD-hard.

CCE via no-regret learning

A distribution μ over joint policies is a CCE if no agent benefits from deviating before observing the correlation device's recommendation. If all agents run no-regret algorithms, the empirical distribution of play converges to the CCE set. The market's efficiency guarantee extends to CCE.

PSRO meta-game

Maintain populations $\mathcal{P}^\alpha = \{\pi_1^\alpha, \ldots, \pi_K^\alpha\}$. Solve the induced finite meta-game for a CCE; train approximate best responses via MAPPO; add to population; repeat. Converges to an approximate CCE of the full POSG.

Policy architecture: multi-agent decision transformer.

- **Library tokens:** formula status, query estimates, dependency counts.
- **Portfolio tokens:** held/posted contract positions, payoff features.
- **Market tokens:** outstanding offers, public ledger state.

Training: MAPPO with CTDE (centralized critic / decentralized actor), population-based training, five-level curriculum ($n \in \{4, 8, 16, 32, 64\}$ theorem pairs), fixed $m = 2$.

Synthetic Training Environment

Proof kernel

Success probability for agent α on formula ϕ with effective budget τ_{eff}:

$$p^{\alpha}(\phi, \tau_{\text{eff}}) = g(\phi)\Big(1 - e^{-\rho^{\alpha}(\phi)(\tau_{\text{eff}}^{\kappa} - \tau_{\text{prev}}^{\kappa})}\Big)$$

- $g(\phi) = 1$ iff ϕ is true and all dependencies already resolved (hard feasibility gate).
- $\rho^{\alpha}(\phi)$ encodes per-agent difficulty and utility-graph support.
- $\kappa \in (0, 1]$: diminishing returns — repeated failed attempts are less informative.
- τ_{prev}: accumulated effort already spent on ϕ (budgets are non-transferable between attempts).

Conjecture kernel

Same Weibull shape, but targets a *ghost* formula $\psi \notin \mathcal{C}_t^{\alpha}$. Success adds $\psi, \neg\psi$ to the agent's private formula set.

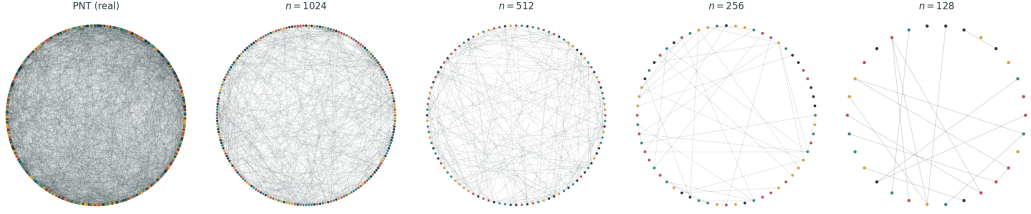
Game initialization

- Formula universe (dependency graph) sampled from the calibrated DAG generator; latent truth assignment drawn at random and dense utility graph for soft dependencies.
- Agents start with only shared axioms visible (empty private libraries beyond L_0^0).
- Each agent wallet initialized with \$1000 and bounty agent seeds long offers on open targets.

Calibrating to Real Lean: Synthetic Graph Generation

Pipeline

1. Parse `.olean` cache to extract the declaration-level dependency DAG of the target repo.
2. Compute rarity scores from dependency frequency; map to difficulty via $d(\phi) = 0.05 + 0.90 \cdot \mathrm{normalized{-}rarity}(\phi)$.
3. Fit a layered, clustered DAG generator to match the real graph's depth, degree distribution, and cross-cluster density.
4. Train the strategy module on synthetic rollouts; deploy zero-shot into Agora prompts.



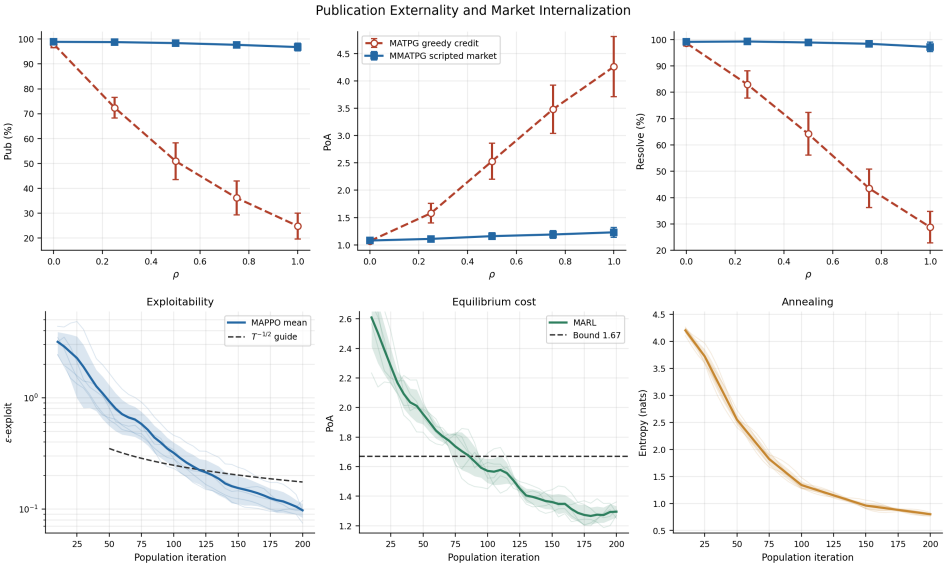
Synthetic Experiments: Main Results

Setup: $n = 8$ theorem pairs, $m = 2$ agents, DAG depth $d = 3$, $\ell = 0.1$, horizon $H = 100$.

Policy	SC	PoA	ε-exploit	Resolve%
OPT (centralized)	27.30 ± 1.98	1.00	—	100.0%
Centralized RLS	27.98 ± 2.18	1.02 ± 0.02	—	100.0%
Greedy credit, no market	79.11 ± 5.38	2.91 ± 0.22	—	50.8%
MAPPO, credit (no market)	72.90 ± 5.1	2.67 ± 0.31	2.41 ± 0.28	55.1%
MARL (PSRO+MAPPO, market)	35.12 ± 2.84	1.29 ± 0.11	0.09 ± 0.03	96.7%

- **Market is the primary driver:** greedy credit stalls at 50.8% resolution; even a scripted market reaches 91.3%.
- **Incentive alignment dominates:** adding MAPPO alone (credit, no market) barely helps — PoA still 2.67.
- **Learning adds on top:** MARL beats scripted market on all four metrics; exploitability 0.09 confirms approximate CCE.

Synthetic Experiments: Externality and Dynamics



Synthetic Experiments: Scaling

Scaling with task graph size n ($m = 2$)

n	PoA	ε-expl.	Resolve%	Time (hr)
4	1.19 ± 0.08	0.06	98.4%	0.6
8	1.29 ± 0.11	0.09	96.7%	1.4
16	1.41 ± 0.14	0.14	93.1%	3.8

PoA rises mildly but stays below the bound. Training scales $\approx O(n^{1.7})$ (attention over formula tokens).

Scaling with agent count m ($n = 8$, $\ell = 0.1$)

m	SC	PoA	Bound	ε-expl.
2	35.12	1.29	1.67	0.09
4	24.81	1.48	1.94	0.16
8	19.34	1.61	2.08	0.24

More agents $\Rightarrow$ lower absolute makespan but higher coordination surface. At $m = 8$: MARL SC 19.34 vs. RLS $\approx$18.5 — **decentralization penalty $\approx$5%**.

Agora: MMATPG on Real Lean 4 Projects

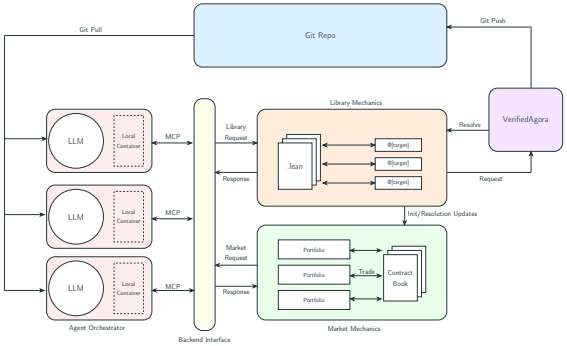


Figure 1: Agora platform architecture. Library Mechanics owns Lean validation; Market Mechanics owns financial invariants; the Backend Interface exposes MCP tools to LLM agents. The Agent Orchestrator provisions containers and seeds the market.

VerifiedAgora

- `@[target]` tags mark proof obligations.
- Solved targets must be `sorry`-free and axiom-clean.
- Submissions compiled, automerged if valid, rebuilt — invalid content rejected without breaking the repo.

MMATPG → Agora

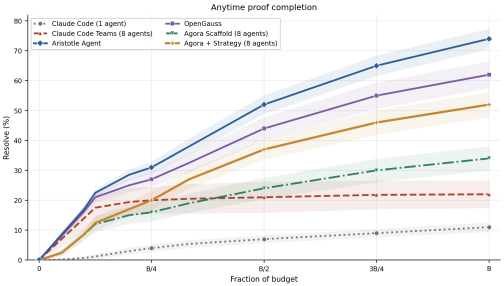
Formula	Lean `Prop` declaration
Library	Git branch (shared/local)
Resolution	VerifiedAgora certified
Publication	Commit to shared branch
Contract	Target-conditioned asset
Proof kernel	LLM + Lean binary oracle

Real-World Evaluation: PNT Benchmark (MiniCTX)

Benchmark: 92 `sorry`'d targets in the PNT Lean formalization. 6,563 total declarations, 27,898 dependency edges, 1-hour wall-clock budget.

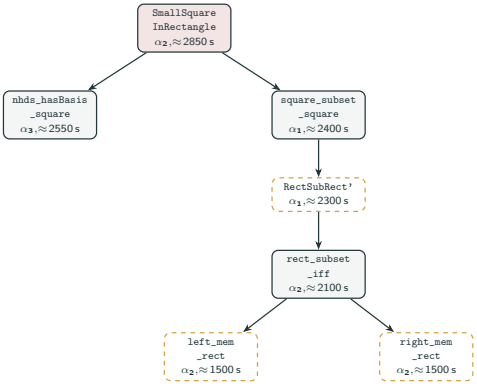
System	Easy	Med.	Hard	**Overall**
A1: Claude Code (1)	21%	5%	0%	11%
A2: Code Teams (8)	38%	12%	0%	22%
A3: Aristotle Agent	90%	68%	42%	74%
A4: OpenGauss	82%	54%	25%	62%
B1: Agora (8)	51%	24%	8%	34%
C1: +Strategy	**72%**	**41%**	**25%**	**52%**

System	Dupl.% ↓	DepUnlock ↑
A2: Teams	36%	2
B1: Agora	21%	5
C1: +Strategy	**8%**	**9**



- Gains concentrate on **Medium and Hard** targets — where dependency-aware allocation matters most.
- Duplication cut from 36% to 8%; dependency unlocks nearly doubled.
- Gap to purpose-built SOTA (Aristotle, 74%) still exists, reflecting prover quality.

Qualitative: SmallSquareInRectangle



Gold dashed = conjectured mid-proof, not original targets.

Time	Agent	Declaration	Kind
≈1500 s	α_2	`left/right_mem_rect`	conjectured
≈2100 s	α_2	`rect_subset_iff`	target
≈2300 s	α_1	`RectSubRect'`	conjectured
≈2400 s	α_1	`square_subset_square`	target
≈2550 s	α_3	`nhds_hasBasis_square`	target
≈2850 s	α_2	`SmallSquareInRectangle`	final target

- **Parallelism:** α_3 works the hard analytic branch ($t \approx 1500$–2550 s) with no coordination needed.
- **Conjecture as decomposition:** 3 conjectured helpers turned difficult downstream proofs into 2–4 line invocations.
- **Publication cascade:** `rect_subset_iff` immediately unblocked α_1's stalled attempt on `RectSubRect'`.
- Market settlement reweighted open offers after each commit.

Future Work

Near-term

- **Domain adaptation:** fine-tune strategy module on real Lean trajectories (currently zero-shot from synthetic). Also pretrain the MAPPO with expert trajectories to speed up convergence.
- **Ablation completion:** Perform more rigorous no-population, no-curriculum, and no-shaping ablations.
- **Broader benchmarks:** evaluate on more MiniCTX repositories beyond PNT. Additionally evaluate for longer and with more agents, if computationally feasible.

Longer-term

- **Centralized regime:** implement and compare to a fully centralized multi-agent theorem proving system.
- **Asynchronous model:** move beyond epoch discretization to token-budget or continuous-time formulation.
- **Cooperative extension:** study collaborative game regime for mixed human–AI formalization teams.
- **PoA theory:** characterize tight equilibrium inefficiency bounds under credit rewards.

Q&A

Listing 1: Helper Lemmas, α_2, $\approx 1500\,\mathrm{s}$

Conjectured leaf lemmas introduced while proving `rect_subset_iff`.

```
lemma left_mem_rect (z w : ℂ) : z ∈ Rectangle z w := by
  apply And.intro (by norm_num) (by norm_num)

lemma right_mem_rect (z w : ℂ) : w ∈ Rectangle z w := by
  apply And.intro;
  · simp [Set.mem_Icc];
  · simp [Set.mem_Icc]
```

Listing 2: First Target, α_2, $\approx 2100\,\mathrm{s}$

`rect_subset_iff`, enabled by the two helper lemmas.

```
lemma rect_subset_iff {z w z' w' : ℂ} :
    Rectangle z' w' ⊆ Rectangle z w ↔ z' ∈ Rectangle z w ∧ w' ∈ Rectangle z w := by
      refine' ⟨ fun h => ⟨ _, _ ⟩, fun h => _ ⟩;
      · exact h ( left_mem_rect _ _ );
      · apply h; exact right_mem_rect z' w';
      · unfold Rectangle at *;
        simp_all +decide [ Set.subset_def, Complex.mem_reProdIm ];
        unfold Set.uIcc at *;
        grind
```

Listing 3: Utility Lemma, α_1, $\approx 2300\,\mathrm{s}$

`RectSubRect'`, conjectured after `rect_subset_iff` became public.

```
lemma RectSubRect' {z₀ z₁ z₂ z₃ : ℂ} (x₀_le_x₁ : z₀.re ≤ z₁.re) (x₁_le_x₂ : z₁.re ≤ z₂.re)
    (x₂_le_x₃ : z₂.re ≤ z₃.re) (y₀_le_y₁ : z₀.im ≤ z₁.im) (y₁_le_y₂ : z₁.im ≤ z₂.im)
    (y₂_le_y₃ : z₂.im ≤ z₃.im) :
    Rectangle z₁ z₂ ⊆ Rectangle z₀ z₃ := by
      apply rect_subset_iff.mpr;
      apply And.intro;
      · apply And.intro;
        · exact Set.mem_uIcc_of_le ( by linarith ) ( by linarith );
        · exact Set.mem_uIcc_of_le y₀_le_y₁ ( by linarith );
      · constructor <;>
          [ exact Set.mem_uIcc_of_le ( by linarith ) ( by linarith )
          ; exact Set.mem_uIcc_of_le ( by linarith ) ( by linarith ) ]
```

Listing 4: Square Monotonicity, α_1, $\approx 2400\,\mathrm{s}$

square_subset_square, enabled by the conjectured RectSubRect' lemma.

```
lemma square_subset_square {p : ℂ} {c₁ c₂ : ℝ} (hc₁ : 0 < c₁) (hc : c₁ ≤ c₂) :
    Square p c₁ ⊆ Square p c₂ := by
      apply RectSubRect';
      all_goals norm_num [ Complex.ext_iff ] at *; linarith
```

Listing 5: Hard Branch, α_3, $\approx 2550\,\mathrm{s}$

`Complex.nhds_hasBasis_square`, the longest proof in this chain.

```
theorem Complex.nhds_hasBasis_square (p : ℂ) : (𝓝 p).HasBasis (0 < ·) (Square p ·) := by
  refine' ⟨ fun s => ⟨ fun hs => _, fun hs => _ ⟩ ⟩;
  · rw [ Metric.mem_nhds_iff ] at hs;
    have h_square_ball : ∀ ε > 0, Square p (ε / 2) ⊆ Metric.ball p ε := by
      intro ε hε;
      intro x hx; rw [ Square_apply p ( half_pos hε ) ] at hx;
      simp_all +decide [ Complex.dist_eq, Complex.normSq ];
      exact Real.sqrt_lt' hε |>.2
        ( by norm_num [ Complex.normSq ] ; nlinarith [ hx.1.1, hx.1.2, hx.2.1, hx.2.2 ] );
    exact ⟨ hs.choose / 2, half_pos hs.choose_spec.1,
      Set.Subset.trans ( h_square_ball _ hs.choose_spec.1 ) hs.choose_spec.2 ⟩;
  · unfold Square at hs;
    obtain ⟨ i, hi, hi' ⟩ := hs;
    refine' Filter.mem_of_superset _ hi';
    rw [ rectangle_mem_nhds_iff ];
    norm_num [ Set.uIoo ];
    exact ⟨
      ⟨ by cases max_cases (-i + p.re) (i + p.re) <;>
           cases min_cases (-i + p.re) (i + p.re) <;> linarith,
        by cases max_cases (-i + p.re) (i + p.re) <;>
           cases min_cases (-i + p.re) (i + p.re) <;> linarith ⟩,
      ⟨ by cases max_cases (-i + p.im) (i + p.im) <;>
           cases min_cases (-i + p.im) (i + p.im) <;> linarith,
        by cases max_cases (-i + p.im) (i + p.im) <;>
           cases min_cases (-i + p.im) (i + p.im) <;> linarith ⟩⟩
```

Listing 6: Final Target, α_2, $\approx 2850\,\mathrm{s}$

`SmallSquareInRectangle`, enabled by both direct parent targets.

```
lemma SmallSquareInRectangle {z w p : ℂ} (pInRectInterior : Rectangle z w ∈ nhds p) :
    ∀ᶠ (c : ℝ) in 𝓝[>]0, Square p c ⊆ Rectangle z w := by
      obtain ⟨ε, hε_pos, hε⟩ : ∃ ε > 0, ∀ c : ℝ, 0 < c → c < ε → Square p c ⊆ Rectangle z w := by
        obtain ⟨ε, hε⟩ : ∃ ε > 0, Square p ε ⊆ z.Rectangle w := by
          have := Complex.nhds_hasBasis_square p;
          exact this.mem_iff.mp pInRectInterior;
        exact ⟨ ε, hε.1, fun c hc₁ hc₂ =>
          Set.Subset.trans ( square_subset_square ( by linarith ) ( by linarith ) ) hε.2 ⟩;
      filter_upwards [ Ioo_mem_nhdsGT hε_pos ] with c hc using hε c hc.1 hc.2
```