Market-Mediated Multi-Agent Theorem Proving in Lean 4

Riyaz Ahuja, Shivansh Gour

Carnegie Mellon University
{riyaza, shivansg}@andrew.cmu.edu

Abstract

Mathematical discovery is not a sequence of isolated proofs but a distributed process across many agents and shared infrastructure. Motivated by this picture, we formalize multi-agent theorem proving as an allocation problem over interdependent proof obligations. We model the decentralized regime as a partially observable stochastic game (MATPG) in which agents privately prove, conjecture, query, and publish Lean-style proof obligations. Under natural credit-based rewards the game admits a publication externality: an agent's reward does not internalize the social value of the public proof events it produces, so capability surplus goes idle whenever the cheap prover for a sequent is not the agent who values its resolution. We show this drives the Price of Anarchy to be unbounded relative to a centralized makespan benchmark. We also introduce the Market-Mediated MATPG (MMATPG), which augments the game with fully collateralized binary contracts on public proof-resolution events; under standard liquidity and resolvability assumptions we recover a $\frac{1}{1-\ell}(2-\frac{1}{m})$ PoA bound. To approximate equilibria of this intractable game, we combine PSRO-style population learning with MAPPO best responses and a transformer policy over library, portfolio, and market tokens. On synthetic theorem-proving games, market-mediated MARL achieves $\mathrm{PoA} = 1.29 \pm 0.11$ and resolves 96.7% of targets, compared to $\mathrm{PoA} = 2.91 \pm 0.22$ and 50.8% resolution for credit-only agents. Deployed through Agora on a Lean 4 Prime Number Theorem benchmark, the learned strategy module raises multi-agent proof completion from 34% to 52% while reducing duplicated work from 21% to 8%. The results support treating multi-agent formalization as strategic coordination over shared formal infrastructure, not parallel proof search.

1 Introduction

In 1896, Hadamard and de la Vallée Poussin independently proved the Prime Number Theorem (PNT): the number of primes at most n is asymptotic to $n/\ln n$. The result had been conjectured nearly a century earlier from numerical evidence by Gauss and Legendre, but its proof required mathematical infrastructure that did not exist at the time, such as Riemann's study of the connection between primes and the zeros ζ, and the subsequent development of contour integration, the theory of entire functions, and analytic continuation. PNT was not proven by directly working on primes, but was rather the eventual payoff of a century of cumulative (and distributed) development of mathematics.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

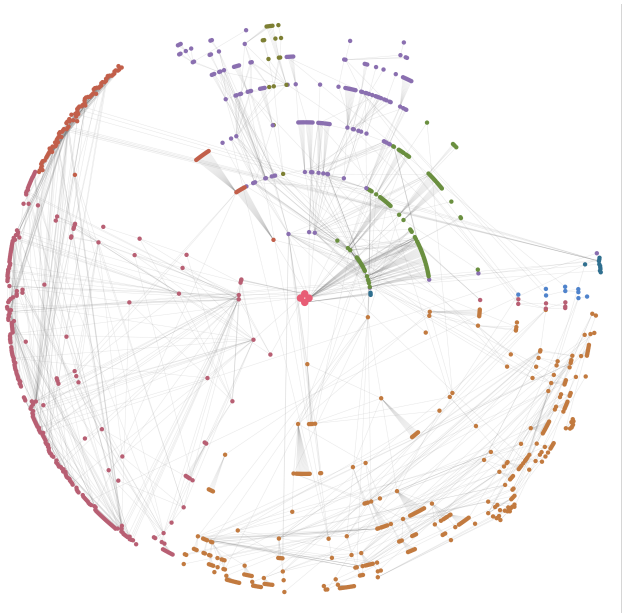

Figure 1: Lean dependency graph for the Prime Number Theorem formalization used as the motivating codebase. Radial distance is dependency depth; angular distance is namespace similarity. The structure illustrates why project-level theorem proving is an allocation problem and not isolated proof search.

This pattern is the rule rather than the exception. Major mathematical results emerge from the cumulative construction of definitions, lemmas, and techniques by communities of researchers. Recent advances mechanize that very dynamic. Proof assistants such as Lean 4 (de Moura and Ullrich 2021), Isabelle (Nipkow, Paulson, and Wenzel 2002), and Rocq (The Coq Development Team 2024) provide kernel-level oracles for correctness; large libraries such as Mathlib (The mathlib Community 2020) provide a shared vocabulary; and large language models (Anthropic 2025) are increasingly deployed as agentic theorem provers that can read source files, inspect goal states, write candidate proofs,

and iterate against the compiler. Yet, existing systems treat each proof obligation in isolation, with each target being handed to a single agent, which itself searches for a proof. Multi-agent structures are rarely used, and when they are, coordination is ad hoc, with difficulties arising from disjoint targets, file-locking, and lack of cooperation within a shared repository.

In contrast, this paper treats proof development as a cooperative process over a shared dependency structure. Figure 1 shows the PNT dependency structure, giving rise to the idea that any system which assigns agents to targets independently misses the central issue. The value of an agent's work depends on when and whether the result becomes public, how it interacts with other agents' private work, and whether other agents can build on it before the budget is exhausted.

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. Within that view, two questions become unavoidable. *What mechanisms align self-interested agents toward the social goal of completing a target set?* And *can MARL find efficient equilibria of those mechanisms?*

Thesis. Credit alone does not align incentives in multi-agent theorem proving. If agents are rewarded only for final targets credited to them, the system fails to route work to the cheapest prover whenever credit and capability are misaligned across agents: capability surplus sits idle and the agent who values a target grinds it themselves. A verifiable contract market over public proof events corrects this by making it so that demand for an intermediate result becomes private reward for whichever agent publishes it, and prices drive the right agents to the right tasks.

Contributions. The paper distills the broader thesis (Ahuja 2026) into the following contributions.

- A sequent-and-library formalism for multi-agent theorem proving, with private and public libraries, dependency-closed publication, and proof / conjecture / query / publish primitives.
- A non-cooperative POSG formulation of the decentralized problem (MATPG), a mis-internalization theorem identifying that credit-based rewards fail to compensate cross-agent value, and a heterogeneous-cost construction showing the equilibrium ratio is unbounded relative to a centralized makespan benchmark.
- The Market-Mediated MATPG (MMATPG), a minimal market mechanism based on collateralized binary claims on public proof-resolution events. Under liquidity and resolvability assumptions we recover a $\frac{1}{1-\ell}(2-\frac{1}{m})$ PoA bound that extends to coarse correlated equilibria.
- A PSRO+MAPPO MARL pipeline with a transformer policy over library / portfolio / market tokens, targeting ε-CCE rather than exact Nash.
- Empirical validation on synthetic games calibrated from the PNT formalization, and a real-Lean deployment via the Agora platform on a 92-target PNT benchmark.

2 Background and Related Work

Formal and neural theorem proving. Lean 4 is a dependent type theory and programming language whose extensible, self-bootstrapped architecture makes it well suited to interactive formalization and metaprogramming (de Moura and Ullrich 2021). Mathlib is its main mathematical library, organizing roughly 1.9M lines of Lean code into a single coherent declaration graph (The mathlib Community 2020). Early neural theorem provers operate at the tactic level: GPT-f (Polu and Sutskever 2020), PACT (Han et al. 2022), and ReProver (Yang et al. 2023) treat each lemma as an isolated tree-search problem and are evaluated on isolated-lemma benchmarks such as MiniF2F (Zheng, Han, and Polu 2021). More recent agentic systems—Aristotle (Achim et al. 2025), Open Gauss (Math, Inc. 2026), Claude Code with Lean LSP access—operate at project scope, reading the repository and iterating on files. The MiniCTX benchmark (Hu, Zhu, and Welleck 2024) was developed to evaluate proof search in full project context. None of these systems provide a structured coordination layer between multiple agents working on the same project; that layer is the topic of this paper.

Algorithmic game theory and mechanism design. The Price of Anarchy (Koutsoupias and Papadimitriou 1999; Roughgarden and Tardos 2002) measures the loss of social efficiency at worst-case equilibrium relative to an optimal centralized policy. Mechanism design (Myerson 1981) asks how to align private incentives with social goals via rules and payments. The failure mode in our setting is a positive externality: an agent's publication creates value for others that is not captured by its own reward. The market used here is a deliberately minimal mechanism: collateralized binary claims on public proof-resolution events. Settlement is verifiable by the proof assistant, not subjective.

MARL and equilibrium learning. Computing exact Nash equilibria of large POSGs is intractable (PPAD-hard even for two-player general-sum normal-form games (Daskalakis, Goldberg, and Papadimitriou 2009)). We use the coarse correlated equilibrium (CCE) as the relaxation, which is naturally connected to no-regret dynamics (Hart and Mas-Colell 2000). Policy-Space Response Oracles (PSRO) (Lanctot et al. 2017) operationalize this idea in extensive-form games by maintaining finite policy populations and a meta-game over them. We use MAPPO (Yu et al. 2022) as the approximate best-response oracle, with centralized training and decentralized execution.

3 Formal Model

We represent mathematical statements syntactically. Let $\mathcal{F}$ be a formula universe equipped with an involution $\neg : \mathcal{F} \to \mathcal{F}$; in a Lean instantiation, formulas are propositions corresponding to theorem declarations. A *sequent* is a proof obligation

$$s = [\Gamma \vdash \phi] \in \mathcal{P}_{\mathrm{fin}}(\mathcal{F}) \times \mathcal{F},$$

where Γ is the antecedent context and ϕ is the consequent formula.

3.1 Libraries

A library state at time $t \in \mathbb{N}$ is a tuple

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

where $\mathcal{C}_t \subseteq \mathcal{F}$ is the set of *concrete* (stated or conjectured) formulas, $\mathcal{RS}_t$ is the set of *resolved* sequents (proof obligations marked solved), $\delta_t : \mathcal{RS}_t \to \mathcal{P}_{\text{fin}}(\mathcal{F})$ records dependencies of the proof witnesses, and $\Theta_t : \mathcal{RS}_t \to \mathbb{N}^+ \times \mathcal{N}$ records solve time and credited agent. Libraries are monotone and respect a negation-symmetry condition: $\phi \in \mathcal{C}_t \iff \neg\phi \in \mathcal{C}_t$, but $[\Gamma \vdash \phi] \in \mathcal{RS}_t \implies [\Gamma \vdash \neg\phi] \notin \mathcal{RS}_t$.

A sequent in $\mathcal{RS}_t$ is locally resolved but may rely on dependencies that are themselves unresolved. We define the *fully resolved* set $\mathcal{FRS}_t \subseteq \mathcal{RS}_t$ inductively: axioms are fully resolved, and a resolved sequent is fully resolved when each of its dependencies has a fully resolved witness in the current library. Publication of a target therefore requires exposing the witnesses needed to check it.

In the multi-agent setting each agent $\alpha \in \mathcal{N}$ maintains a private library $\mathcal{L}_t^\alpha$, and the environment maintains a public library $\mathcal{L}_t^0$. Public knowledge is included in every private view: $\mathcal{L}_t^0 \subseteq \mathcal{L}_t^\alpha$ for every α, t.

3.2 Action Primitives

Agents interact with the libraries through four non-market primitives, summarized in Table 1.

Table 1: Core action primitives.

Action	Meaning
$\text{Prove}(s, \tau)$	Attempt sequent s for budget τ
$\text{Conj}(s, \tau)$	Propose helper formula anchored at s
$\text{Pub}(C, R)$	Publish dependency-closed results
$\text{Qry}(s)$	Estimate success probability and time
Market action	Create / accept / cancel offers (Section 5)
$\perp$	No-op / wait

Proof and conjecture actions are durative stochastic jobs. Agent α's proof kernel maps a library, a target, and a budget to a Bernoulli outcome $(0, \perp)$ or a success $(1, d)$ with a dependency set $d \subseteq \Gamma$. The conjecture kernel similarly proposes a new ghost formula. Publication is deterministic and dependency-closed: publishing a target also publishes the transitive closure of its dependency witnesses, the formal counterpart of submitting a Lean proof together with the helper declarations needed for it to compile. Full kernel signatures are deferred to Appendix B (p. 15).

3.3 Centralized Baseline

The centralized Multi-Agent Theorem Proving problem (MATP) is the POMDP in which a single planner controls all agents. The state ω_t at time t records the public library $\mathcal{L}_t^0$, every agent's private library $\mathcal{L}_t^\alpha$, the active proof and conjecture jobs (target, remaining budget, accumulated effort), and an unobserved *latent dependency structure* δ^* that determines feasibility. The transition function decomposes by action type. *Prove*/*Conj* actions tick their job counters and—when a job completes—draw a Bernoulli outcome from the proof kernel, on success adding the resolved sequent to the acting agent's private $\mathcal{RS}^\alpha$ together with its dependency witnesses. *Pub* moves a dependency-closed subset of $\mathcal{RS}^\alpha$ into $\mathcal{RS}^0$, making the witnesses public. *Qry* draws a noisy estimate from the query model. Latent dependencies are revealed as their consuming sequents are attempted, so the planner observes a strictly increasing approximation to δ^* over time.

The planner seeks to minimize the expected makespan

$$C_{\max}(\pi) = \mathbb{E}^\pi\big[\inf\{t \geq 0 : \Phi^* \subseteq \mathcal{FRS}_t^0\}\big]$$

for a target set Φ^*. The problem is NP-hard even with deterministic processing times (it contains $P\|C_{\max}$ scheduling (Garey and Johnson 1979)).

POMDP → MDP via PPI. Although δ^* is unobserved, the centralized planner can act on the *partially perfect-information* (PPI) sufficient statistic: the union of all currently revealed dependency information, public and private. Under the closure assumptions of Section 3.1 (p. 3), this statistic is a sufficient summary of ω_t for the makespan objective—two histories with the same PPI state induce the same conditional distribution over future $C_{\max}$. The centralized POMDP therefore reduces to an MDP over PPI states, a fact we use to argue centralized scheduling guarantees apply in expectation. A formal statement and proof are in Appendix A (p. 15).

RLS as the reference policy. We use Reactive List Scheduling (RLS) as the centralized reference: maintain a queue of known unresolved tasks, assign idle agents to highest-priority available tasks, and add newly revealed dependencies as proofs complete. By an analogue of Graham's list-scheduling argument (Graham 1966)—with informational precedence in place of operational precedence—

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

RLS is not the focus of this paper; it provides the denominator for efficiency comparisons in the decentralized game. The full derivation is in Appendix C (p. 15).

4 The Decentralized Game

The decentralized game is the centralized MATP POMDP minus the single-controller assumption, plus three additions: per-agent observation kernels, individual reward functions, and a market state that we add in Section 5 (p. 4). State, action types, and transition dynamics are inherited from the centralized model; what changes is who controls each action, what each controller can see, and what each controller is trying to maximize.

4.1 MATPG as a POSG

Definition 1 (MATPG)**.** *The* Multi-Agent Theorem Proving Game *is the partially observable stochastic game*

$$\mathcal{G}_{\text{MATPG}} = \big(\mathcal{N}, \mathsf{S}, \{\mathcal{A}^\alpha\}, \{\mathsf{O}^\alpha\}, T, T^0, \{Z^\alpha\}, \{r^\alpha\}, \gamma\big),$$

where S *and the transition kernel* T *are those of the centralized* MATP. *At each step every agent independently selects an action from* $\mathcal{A}^\alpha = \{\mathrm{Prove}, \mathrm{Conj}, \mathrm{Pub}, \mathrm{Qry}, \bot\}$ *subject to admissibility, and* T *applies the corresponding transition components in parallel (with conflict-resolution rules for simultaneous publication of the same target, given in Appendix A (p. 15)). The new ingredients are the per-agent observation*

$$o_t^\alpha = (\mathcal{L}_t^\alpha, \Omega_t^\alpha, \mathcal{L}_t^0),$$

giving each agent only its own private library, its own query responses, and the public library, and per-agent rewards r^α *replacing the social makespan objective. Other agents' private libraries, active jobs, and cumulative effort are hidden.*

Two observations follow. First, the underlying state space and dynamics match the centralized POMDP, so any policy that the centralized planner could implement remains feasible in the decentralized game (it just requires coordination that no single agent unilaterally provides). Second, the PPI sufficient statistic of Section 3.3 (p. 3) is no longer available to any one agent: the planner's PPI state is the union of agent private states, but each agent observes only its own slice. Decentralization loss therefore arises from two sources: the partial information each agent has, and the misalignment between individual and social objectives. The full POSG specification (state space, transition kernel, observation kernel) is in Appendix A (p. 15). The social cost remains expected makespan; we measure decentralization loss by the Price of Anarchy:

$$\mathrm{SC}(\pi) = \mathbb{E}^\pi\big[C_{\max}\big], \qquad \mathrm{PoA} = \frac{\sup_{\pi^* \in \mathrm{NE}} \mathrm{SC}(\pi^*)}{\inf_\pi \mathrm{SC}(\pi)}.$$

For learned policies we use approximate CCE in place of exact Nash; the same social cost is used for evaluation.

4.2 Credit Rewards and the Publication Externality

A natural individual reward gives credit for new fully resolved public sequents:

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

This mirrors academic priority: agents are paid only for targets credited to them after publication. Credit rewards depend only on the publisher's own contributions; they do not pay for the social value other agents derive from those contributions, so the reward function fails to internalize the cross-agent value of public proof events.

Theorem 2 (Mis-internalization)**.** *In any* MATPG *with* $|\mathcal{N}| \geq 2$ *and credit-based rewards, if some sequent* s *has cross-agent value—there exist agents* $\alpha \neq \beta$ *with* α *a feasible prover and* $v_\beta(s) > 0$*, while* $v_\alpha(s) = 0$*—then there exist states at which publishing (or proving and publishing)* s *strictly increases* β*'s expected future return and weakly decreases* α*'s. As a consequence, no own-contribution reward can match the social marginal value of public resolution.*

Proof sketch. Suppose at time t, α either has s privately resolved or can resolve it cheaply, and β's target depends on s. Publishing s expands β's feasible set and increases β's expected return; the action costs α one timestep of effort and earns α zero credit. The full argument and a discrete-time formulation are in Appendix D (p. 15). □

Theorem 2 (p. 4) has two equilibrium consequences. *(a) Capability misallocation*: when α is the cheap prover for s but the credit accrues to β, α has no own-credit incentive to take on s, so β pays the full cost itself and capability surplus goes idle. *(b) Intermediate hoarding*: an agent who has obtained a partial helper as a byproduct may delay its publication when this serves their own credit calculus on a downstream target. We show below that mode (a) is sufficient to drive the equilibrium ratio to infinity; mode (b) is recorded in Appendix D (p. 15) and observed empirically.

Unbounded PoA **from capability misallocation.**

Theorem 3 (Unbounded PoA without market)**.** *The Price of Anarchy of the* MATPG *with credit-based rewards is unbounded: for every* $M > 0$ *there exists an instance with* $\mathrm{PoA} > M$.

Proof sketch. Take two agents α, β and two sequents with $\delta^*(s_0) = \{\phi_h\}$, $\Phi^* = \{s_0\}$. Set credit values $v_\beta(s_0) = 1$ and all others zero, and per-step success probabilities so that expected proof times are $c_\alpha(s_h) = 1$, $c_\beta(s_h) = K$, $c_\alpha(s_0) = K$, $c_\beta(s_0) = 1$. Under credit rewards α has zero incentive to attempt s_h (no credited target depends on it for α), so α allocates effort to its own private credited targets. β is the only agent who values s_0, so β proves s_h first at expected cost K and then s_0 at cost 1, giving social cost $K + 1$. The centralized planner assigns s_h to α and s_0 to β in parallel and resolves the target in expected time $O(1)$. Letting $K \to \infty$ drives $\mathrm{PoA} \to \infty$. Under the contract market, β posts a long offer on s_h at price $p < 1 - \ell - 1/q$; α accepts, proves at cost 1, and collects $1 - \ell$ per unit, restoring parallel execution. A full discrete-time proof, the analogous hoarding-mode construction, and the chain generalization are in Appendix D (p. 15). □

The pathology is allocational, not strategic nondisclosure: an agent who has independently proved every link of a credited chain still publishes its final target to claim the credit, but credit rewards provide no instrument for routing work between agents whose costs and credit-values disagree, so capability surplus is left idle. More learning alone cannot repair this, since a competent learner trained against misaligned credit rewards will simply learn the misaligned behavior more efficiently. What is missing is a price.

5 Market-Mediated Theorem Proving

We now augment the MATPG with a market mechanism whose settlement is tied to public proof events. The mechanism is intentionally minimal: it is the smallest market needed to price the publication externality, not an attempt to design a complete economy for mathematics.

5.1 Collateralized Bilateral Contract Market

A *contract type* is a triple $\chi = (s, T, \ell) \in \mathcal{S} \times \mathbb{N} \times [0, 1]$, representing a normalized unit-notional claim on the event

$$E_{s,T} := \mathbf{1}[s \in \mathcal{RS}_T^0]$$

that sequent s is publicly resolved by deadline T, with loss parameter ℓ. The two complementary positions have terminal payoffs

$$\Pi(\chi^+) = \begin{cases} 1-\ell, & E_{s,T} = 1, \\ -\ell, & E_{s,T} = 0, \end{cases} \qquad \Pi(\chi^-) = -\Pi(\chi^+).$$

The long side χ^+ profits when s is publicly resolved; the short side χ^- profits otherwise.

Trading, collateral, and settlement. New contracts are issued as matched long/short pairs so no net wealth is created. Agents post, accept, or cancel offers; a market action is admissible only if the agent's minimum marked balance (cash minus worst-case mark on every held position and posted offer) stays nonnegative. Full collateralization eliminates default at the cost of constraining demand from capital-poor agents; we seed initial liquidity through an exogenous *bounty agent* that posts long offers on open targets. At each t, contracts with deadline $T = t$ settle against the *public* library $\mathcal{RS}_t^0$: an agent who proves s privately but does not publish has $s \in \mathcal{RS}_t^\alpha$ but $s \notin \mathcal{RS}_t^0$ and does not collect. This is the mechanism's only point of contact with the proof events, and it is what makes the contract a price on public resolution rather than on private claim.

5.2 Market-Mediated MATPG

Definition 4 (MMATPG)**.** *The* Market-Mediated MATPG*, $\mathcal{G}_{\text{MMATPG}}$, extends $\mathcal{G}_{\text{MATPG}}$ with a market state $\mathcal{M}_t = (\{\mathcal{P}_t^\alpha\}_\alpha, \mathcal{ML}_t)$ comprising each agent's portfolio and the public order book, market actions, and the* balance-change *reward*

$$r_{\text{mkt}}^\alpha(w_t, a_t) := B_{t+1}^\alpha - B_t^\alpha,$$

where B_t^α is the canonical valuation of α's portfolio.

The balance-change reward telescopes to terminal-minus-initial balance up to discounting, and unifies trading profits, proof effort, and settlement payoffs in a single quantity. The full extension of state, observation, and transition kernels is given in Appendix A (p. 15).

5.3 Internalization and Bounded PoA

The market gives the externality of Theorem 2 a price.

Theorem 5 (Market internalization)**.** *Suppose agent β values public resolution of sequent s by deadline T at $v_\beta > 0$, and agent α can resolve s at expected cost c_α. If $v_\beta > c_\alpha$, the contract market admits trades for which prove-and-publish is privately profitable for α and individually rational for β. Settlement reads the public library, so private withholding cannot collect.*

Proof sketch. β takes long exposure on s at unit price $p < v_\beta$; α acquires long exposure (by purchase or matched issue) at $p < 1 - \ell - c_\alpha/q$ for sufficient quantity, proves s, publishes, and collects $1 - \ell$ per unit. The deviation has positive surplus; the settlement check on $\mathcal{RS}_T^0$ forces publication. □

The argument upgrades to an equilibrium efficiency bound under transparent assumptions.

Theorem 6 (Bounded PoA for MMATPG)**.** *Suppose (i) all agents have initial capital at least a threshold depending on the instance (sufficient liquidity); (ii) every required sequent is resolvable by some agent with positive probability; (iii) for every required sequent the gain from trade is positive. Then every Nash equilibrium of $\mathcal{G}_{\text{MMATPG}}$ resolves every required sequent in finite expected time, and*

$$\text{PoA} \le \frac{1}{1-\ell}\left(2 - \frac{1}{m}\right).$$

Proof sketch. If a required sequent is unpublished at equilibrium, an agent has a profitable unilateral deviation: take long exposure, prove, publish, settle. Once publication is incentivized, the remaining allocation inefficiency is bounded by the centralized RLS ratio; the factor $1/(1-\ell)$ accounts for contract loss/friction. As $\ell \to 0$ the bound approaches the centralized $(2 - 1/m)$. A full proof, and the matching CCE-version used below, appear in Appendix E (p. 16). □

This bound is not tight on every instance. Its role is qualitative: the credit-only MATPG can have arbitrarily inefficient equilibria, while the MMATPG inherits a scheduling-style bound.

What the market communicates. Beyond settlement, the order book is itself a coordination signal: long offers reveal demand for specific targets, and existing positions are public, so the cheap prover for a sequent can see who values it without relying on private observations. With respect to mechanism-design, the market converts the cross-agent value identified by Theorem 2 into private payoff, in the same way Pigovian instruments price other externalities (Pigou 1920).

6 Learning Approximate Equilibria

The state of the MMATPG includes private libraries, query histories, durative proof jobs, market ledgers, and portfolios. Exact Nash computation is PPAD-hard already in the underlying credit game, so we instead target *approximate coarse correlated equilibria*.

6.1 Why CCE

We target the coarse correlated equilibrium (CCE) of the MMATPG, the weakest standard equilibrium concept reachable by no-regret dynamics (Hart and Mas-Colell 2000). Two properties make this the natural choice for our setting. First, exact Nash is PPAD-impossible at this scale; CCE is polynomial-time accessible. Second, the bounded-PoA argument of Theorem 6 extends to CCE without modification: the prove-and-publish deviation is unconditional—profitable against any opponent mixture—so ε-CCE inherits the $\frac{1}{1-\ell}(2 - \frac{1}{m})$ bound up to an additive $O(\varepsilon)$ slack (Appendix E (p. 16)).

6.2 Policy Architecture

Each agent is parameterized by a transformer-style actor (Vaswani et al. 2017) with factored output heads. Variable-length token sets carry the observation:

- **Library tokens**: per concrete formula, status bits (local resolved, public resolved, in-progress), difficulty, dependency counts, cumulative effort, query estimates, and the negation-pair sign.
- **Portfolio tokens**: per held position, contract parameters, quantity, marked payoff features, and worst-case liability.
- **Market tokens**: per outstanding offer, target, side, deadline, quantity, price, ownership flags.

The actor outputs an action type, a target pointer over candidate entities, and action parameters; an admissibility mask zeros out illegal actions. We use centralized training and decentralized execution (CTDE) (Lowe et al. 2017): during training the critic receives privileged aggregate state; during execution the actor uses only the agent's observation history.

6.3 PSRO + MAPPO Training

Training alternates between population management and best-response learning. For each agent α we maintain a policy population $\mathcal{P}^\alpha = \{\pi_1^\alpha, \dots, \pi_K^\alpha\}$, initialized with scripted heuristics. At each iteration:

1. Estimate the empirical meta-game payoff matrix over population combinations.
2. Compute an approximate meta-strategy CCE via regret matching.
3. Train an approximate best response with MAPPO against the opponent mixture, using GAE advantages (Schulman et al. 2015) and a clipped PPO objective (Schulman et al. 2017).
4. Add the resulting policy to the population.

Under a no-regret meta-strategy algorithm and an $\varepsilon_{\mathrm{BR}}$-approximate best-response oracle, the induced policy distribution after T iterations is an $\tilde{O}(1/\sqrt{T}) + \varepsilon_{\mathrm{BR}}$-CCE of the full MMATPG, and the bounded-PoA bound extends with the same additive slack. The full chain (meta-game CCE $\to$ full-game ε-CCE $\to$ ε-CCE PoA) is in Appendix F (p. 16). In practice $\varepsilon_{\mathrm{BR}}$ is measured by fine-tuning unilateral deviators against fixed opponents—the empirical exploitability metric reported below.

7 Synthetic Environment

We instantiate the MMATPG as a finite, formula-indexed simulation that preserves the core dynamics while making training tractable. We describe the key design choices here; full kernel and architecture details are in Appendix G (p. 17).

7.1 Formula Universe and Graphs

The formula space is $|\mathcal{F}| = 2n$ integer-indexed formulas organized as n theorem pairs; for each pair a latent truth assignment selects one formula as true. False formulas may be concrete and may be attempted, but the proof kernel assigns them zero success probability. True formulas form a hard-dependency DAG (feasibility constraints), and a dense utility graph supplies soft support weights that affect proof and conjecture rates without changing feasibility. Sequents are compressed to formula targets under the canonical local context of the agent's currently resolved formulas, keeping the action space linear in $|\mathcal{F}|$ rather than exponential in antecedent sets.

7.2 Kernels

The proof kernel uses a conditional Weibull form. For agent α attempting formula ϕ with new budget τ after accumulated effort τ_{prev},

$$p^\alpha_{\mathrm{proof}}(\phi, \tau) = g(\phi)\Big(1 - \exp\big[-\rho^\alpha(\phi)((\tau_{\mathrm{prev}} + \tau)^\kappa - \tau_{\mathrm{prev}}^\kappa)\big]\Big).$$

The gate $g(\phi)$ is one only when ϕ is true and all hard dependencies are locally resolved; the rate $\rho^\alpha(\phi)$ falls with difficulty and rises with utility-weighted support; $\kappa \in (0, 1]$ controls diminishing returns. The conjecture kernel shares the shape but introduces a new ghost formula and its negation. The query model is an oracle with persistent per-formula noise: it returns noisy probability and duration estimates given the latent graph and kernel parameters. The market is the collateralized posted-offer book of Section 5, with the bounty agent seeding long offers at initialization.

7.3 Calibration to Real Lean

For Lean transfer we calibrate the synthetic generator to a real repository. We parse built Lean artifacts to extract the declaration-level dependency DAG, compute dependency-frequency rarity scores, map rarity to difficulty via $d(\phi) = 0.05 + 0.90 \cdot \mathrm{normalized\_rarity}(\phi)$, and fit a layered, clustered DAG generator that preserves the real graph's depth, degree distribution, and cross-cluster density. Figure 2 shows the extracted PNT graph and synthetic samples.

7.4 Curriculum and Reward Shaping

The policy is trained with a curriculum over graph size, dependency complexity, and market activity. Reward has two parts. The economically meaningful component is the realized cash delta after settlement and market actions; small shaping terms penalize idle behavior, gas-like action costs, and reward proof/query alignment during training. *All reported social metrics are independent of the shaping terms.*

8 Synthetic Experiments

Unless otherwise noted, experiments use $m = 2$ agents, $n = 8$ theorem pairs, DAG depth $d = 3$, horizon $H = 50$, loss parameter $\ell = 0.1$, and 50 evaluation rollouts. We report expected social cost SC, PoA relative to centralized OPT, empirical exploitability ε-exploit, publication rate, and target resolution rate.

8.1 Baselines

Five policies isolate three sources of improvement (centralized scheduling, incentive alignment, learned coordination): **OPT** (exact centralized DP, available only at small

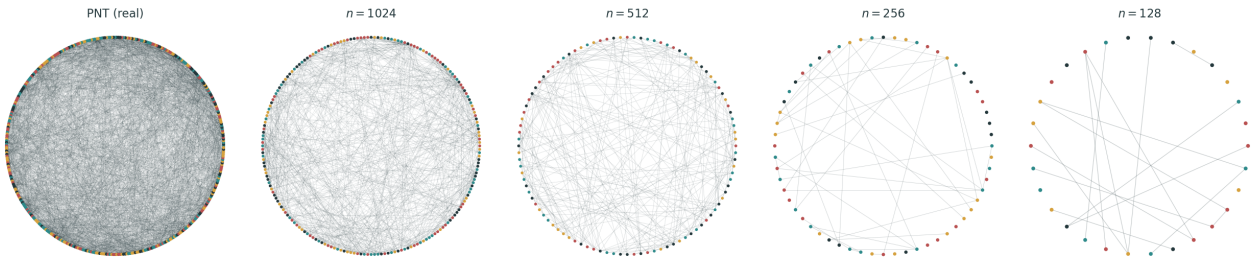

Figure 2: Calibration pipeline. Left: dependency graph of the PNT formalization extracted from `.olean` caches. Right: synthetic graphs sampled at smaller scales preserving depth, clustering, hub structure, and cross-cluster dependency density. The same rarity-to-difficulty mapping is used for real and synthetic instances.

n); **RLS** (centralized reactive list scheduler); **Greedy credit** (each agent greedily maximizes own credit reward, no market); **MAPPO, credit, no market** (the same learning machinery as the full model, but without market actions and with credit reward); **Scripted market** (hand-written heuristic that accepts available bounties, proves high-value targets, publishes immediately); **PSRO + MAPPO market** (the full method).

8.2 Centralized Scheduling Sanity Check

Table 2: RLS approximation ratio and lower-bound tightness. Worst-case bound is 1.50 for $m = 2$ and 1.75 for $m = 4$. All empirical ratios are well below the bound.

		$m = 2$		$m = 4$	
n	d	SC/OPT	OPT/LB	SC/OPT	OPT/LB
4	1	1.040 ± 0.109	1.004 ± 0.043	1.000 ± 0.000	1.000 ± 0.000
4	3	1.027 ± 0.091	1.009 ± 0.060	1.000 ± 0.000	1.000 ± 0.000
8	1	1.032 ± 0.048	1.033 ± 0.031	1.100 ± 0.134	1.075 ± 0.145
8	3	1.066 ± 0.088	1.021 ± 0.024	1.011 ± 0.052	1.015 ± 0.067
8	5	1.049 ± 0.070	1.015 ± 0.017	1.004 ± 0.025	1.008 ± 0.049
16	3	1.035 ± 0.034	1.006 ± 0.005	1.099 ± 0.091	1.076 ± 0.083
16	5	1.036 ± 0.037	1.007 ± 0.014	1.063 ± 0.082	1.047 ± 0.078

Table 2 confirms the centralized reference: across $n \in \{4, 8, 16\}$, depths $d \in \{1, 3, 5\}$, and $m \in \{2, 4\}$, RLS stays well inside the $(2 - 1/m)$ bound. This validates the denominator of our PoA comparisons; the centralized planner is not an unattainable ideal but a near-optimal scheduling reference.

8.3 Externality and Market Internalization

Figure 3 sweeps cross-agent dependency density ρ, the fraction of latent dependencies that connect sequents valued by different agents. As ρ rises, more proof obligations have the cheap prover and the value-holder on opposite sides of an agent boundary, which is exactly the regime where Theorem 3 predicts capability misallocation. The empirical numbers track that prediction: under credit rewards, publication rate, PoA, and resolution all degrade sharply with ρ. Under the market, prices internalize the cross-agent value: the value-holder posts demand, the cheap prover accepts, and execution recovers. This shows the market's effect, as the coordination failure here is an incentive problem, as opposed to a search problem.

Figure 4 sweeps ℓ. Empirical PoA tracks the predicted direction: more friction means weaker incentives. The observed PoA stays below the worst-case bound, which is expected because the bound is worst-case.

8.4 Headline Result and Ablation

Table 3 summarises the headline comparison. The ordering is the most important observation:

$$\text{credit + learning} \ll \text{market + script} < \text{market + learning}.$$

Greedy credit performs poorly (PoA $= 2.91$, 50.8% resolution), and agents withhold byproducts. MAPPO without the market does not solve the issue. It improves SC slightly, but stays at PoA $= 2.67$, exploitability 2.41, and 55.1% resolution, implying learning alone is not enough when the reward is structurally misaligned.

The market is the primary reason for improvement. A scripted market policy already reaches PoA $= 1.45$ and 91.3% resolution; PSRO+MAPPO over the market improves further to PoA $= 1.29 \pm 0.11$, exploitability 0.09 ± 0.03, and 96.7% resolution. Learning adds value, but only on top of incentive alignment. This is the central empirical case for combining mechanism design with MARL rather than treating multi-agent theorem proving as a generic MARL benchmark.

The market should not be read as a replacement for planning but as a decentralized way of making parts of the planning problem visible. A centralized planner can assign an agent to prove a helper because it knows the helper's downstream value. In the decentralized formulation that downstream value is distributed across other agents' private goals; a long offer on a target is a compact public signal that some participant values its resolution, and taking the offer and proving the target converts that value into private reward—the publication externality becomes a price. This is why even the scripted market baseline already performs well: the incentive surface has changed. The learned strategy module then improves on the scripted rule by learning when to pursue dependency chains, when to use redundant proof attempts on bottlenecks, and when to avoid low-probability contracts.

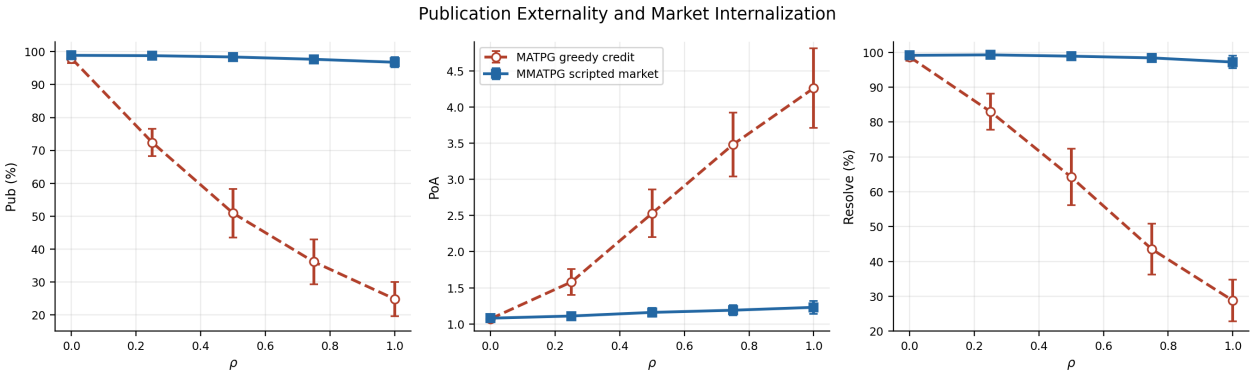

Figure 3: Mis-internalization (credit-only, dashed) versus market resolution (MMATPG, solid) across cross-agent dependency density ρ. As ρ rises, more sequents have a cheap prover and a value-holder on opposite sides of an agent boundary. Left: under credit rewards the cheap prover has no incentive to take on the work, so the public publication rate collapses; under the market, prices route the work and publication stays near 100%. Middle: PoA grows unboundedly without the market; the market holds it below 1.25. Right: resolution rate collapses under credit; the market maintains near-complete resolution.

Table 3: Policy comparison at convergence ($n = 8, m = 2, d = 3, \ell = 0.1$). Theoretical MMATPG bound: $(2-1/m)/(1-\ell) = 1.67$.

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 ± 5.10	2.67 ± 0.31	2.41 ± 0.28	55.1%
Scripted market	39.47 ± 3.21	1.45 ± 0.14	0.82 ± 0.11	91.3%
PSRO+MAPPO, market	$\mathbf{35.12 \pm 2.84}$	$\mathbf{1.29 \pm 0.11}$	$\mathbf{0.09 \pm 0.03}$	$\mathbf{96.7\%}$

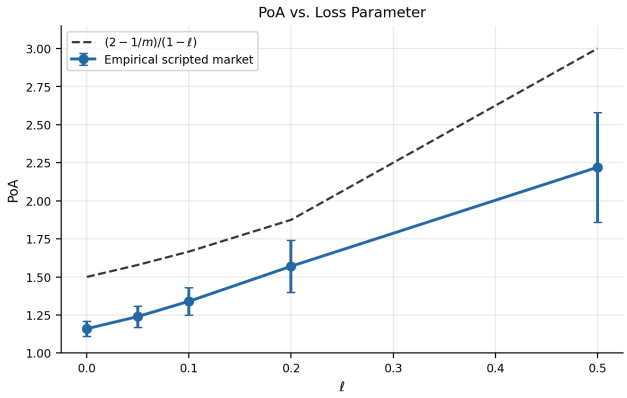

Figure 4: Empirical PoA as a function of the loss parameter ℓ, compared with the theoretical bound $(2 - 1/m)/(1 - \ell)$. Higher friction worsens the bound and the observed outcome; empirical PoA remains below the bound throughout.

8.5 Training Dynamics and Equilibrium Quality

Figure 5 shows the training trajectory. Empirical exploitability decreases toward the approximate-CCE regime at the $O(1/\sqrt{T})$ rate predicted by the convergence theorem; PoA stays below the market bound; entropy annealing prevents premature collapse and allows the meta-strategy to concentrate on robust policies.

Figures 6–7 confirm that the learned behavior is not reward hacking. Publication and market participation move in the directions one would expect of agents that have internalized the contract market: more publishing, more deliberate trading, less idle time; the meta-strategy first uses policy diversity then concentrates.

8.6 Scaling

Table 4: Scaling with task graph size n ($m = 2$, $d = 3$, $\ell = 0.1$).

n	PoA	ε-exploit	Resolve%	Time (hr)
4	1.19 ± 0.08	0.06 ± 0.02	98.4%	0.6
8	1.29 ± 0.11	0.09 ± 0.03	96.7%	1.4
16	1.41 ± 0.14	0.14 ± 0.05	93.1%	3.8

Table 5: Scaling with agent count m ($n = 8$, $d = 3$, $\ell = 0.1$). The theoretical bound is $(2 - 1/m)/(1 - \ell)$.

m	SC	PoA	Bound	ε-exploit
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

Tables 4–5 report scaling. With more formulas, exploitability grows mildly and resolution drops slightly, but PoA stays below the bound. With more agents, absolute makespan falls sharply (more parallelism) while PoA grows toward the bound (more coordination surface). At $m = 8$, SC $= 19.34$ vs. centralized RLS at ≈ 18.5: the market plus strategy module coordinate eight agents to within roughly 5% of a centralized scheduler, with no planner.

9 Lean Deployment via Agora

We now ask whether the abstract framework transfers to real formal mathematics. Agora is a platform that instantiates the MMATPG over real Lean 4 codebases, with verified publication and on-chain-style market settlement. This section describes the mapping; full architecture details and the end-to-end platform diagram are in Appendix H (p. 17). At a high level, containerized LLM agents communicate via MCP with a Backend Interface that delegates Lean validation to Library Mechanics and financial invariants to Market Mechanics, with VerifiedAgora adjudicating which submissions are accepted into the public Git repository.

9.1 Agora as an MMATPG Instantiation

Table 6: Mapping from the abstract MMATPG model to the Agora Lean deployment.

MMATPG **component**	**Agora / Lean implementation**
Formula ϕ	Lean declaration of type `Prop`
Target set Φ^*	Declarations tagged `@[target]`
Private library $\mathcal{L}^\alpha$	Agent workspace / local git branch
Public library $\mathcal{L}^0$	Shared branch after verified merges
Resolved sequent	Target accepted by VerifiedAgora
Publication	Commit accepted contribution to shared branch
Proof kernel	LLM agent + Lean compiler / LSP feedback
Query model	LLM or strategy-module success/time estimate
Contract $\chi = (s, T, \ell)$	Target-conditioned asset with deadline, price
Settlement event	Public sorry-free target resolution by deadline

The mapping is given in Table 6. Each Lean declaration of type `Prop` is a formula; declarations tagged `@[target]` are proof obligations; each agent's workspace is its private library; the shared branch is the public library; and a target counts as resolved when it passes the VerifiedAgora pipeline.

Verification. VerifiedAgora is a Lean library that provides the trusted verification layer. Solved targets must be sorry-free and axiom-clean (depending only on `propext`, `Quot.sound`, and `Classical.choice`). Submissions are compiled, target-matched, automerged when valid, and the project is rebuilt; invalid submissions are rejected without breaking the shared branch. This makes settlement trustworthy: the event underlying a contract is not a subjective judgment but a verified public state transition.

Library and market mechanics. Library Mechanics owns project repositories, target extraction, validation, and merging, and serializes concurrent contributions to avoid breaking downstream builds. Market Mechanics owns wallets, offers, target statuses, and settlement, enforcing money

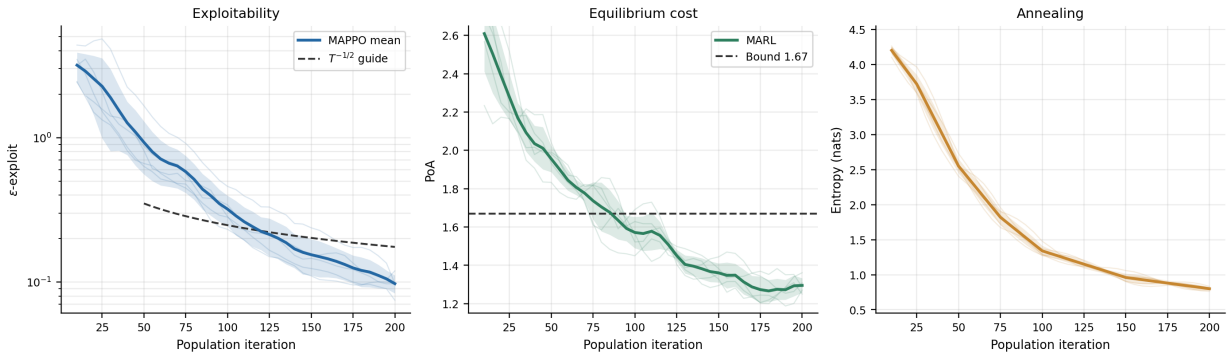

Figure 5: PSRO+MAPPO training dynamics. Left: empirical exploitability tracks the $O(1/\sqrt{T})$ no-regret reference. Middle: PoA converges below the market bound of 1.67. Right: entropy anneals without immediate collapse.

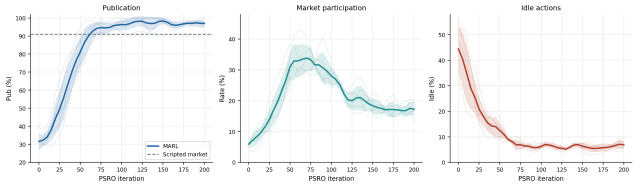

Figure 6: Rollout behavior over PSRO training. Publication rate rises as agents internalize market incentives; market participation overshoots and then becomes more deliberate; idle rate falls as agents learn productive task sequences. The externality is being learned away, not merely suppressed.

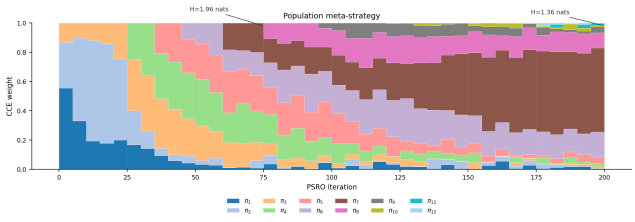

Figure 7: Population meta-strategy over PSRO iterations. Weights spread across diverse best responses in the middle phase, then reconcentrate on robust policies as the empirical meta-game converges.

conservation and full collateralization. When a target resolves, it simplifies all referencing positions and offers and converts determined payoffs into wallet balances.

Agent orchestrator. Each agent runs in a container with MCP-based access to library tools, market tools, Lean LSP tools, and a filesystem. The orchestrator maintains structured agent state (active project, target, wallet, build status, recent market events, suggested actions) and reengages the LLM after every response. The strategy module's recommendations are inserted into the prompt; observed adherence is roughly 85%.

9.2 Training-to-Inference Pipeline

Training directly on Lean is too expensive: each proof attempt costs minutes of LLM inference and Lean compilation, and a single repository offers a fixed target set. We instead train on synthetic MMATPG instances calibrated from the PNT repository (Section 7) and deploy the trained strategy module zero-shot through Agora prompts.

This soft deployment is done so that the synthetic policy is trained on abstract formula ids and stochastic kernels, whereas the real agent sees Lean files, theorem statements, tactic states, import errors, and natural-language context. Forcing the policy output as a hard action would make the training-to-inference gap too brittle for our case. Recommending an action lets the strategy module guide allocation while preserving the prover's ability to react to local proof information.

9.3 PNT Benchmark

We evaluate on the MINICTX Prime Number Theorem formalization (Hu, Zhu, and Welleck 2024): 92 sorry-tagged targets in a repository of 6,563 declarations and 27,898 internal dependency edges. We bucket targets by structural difficulty (Easy/Medium/Hard) using the same rarity-to-difficulty mapping as in the calibration pipeline. All systems receive a one-hour wall-clock budget per run; statistics in Table 7.

We compare two Agora systems against four external baselines. **A1: Claude Code (1)** (Anthropic 2025) is a single agent with Lean LSP access. **A2: Claude Code Teams (8)**

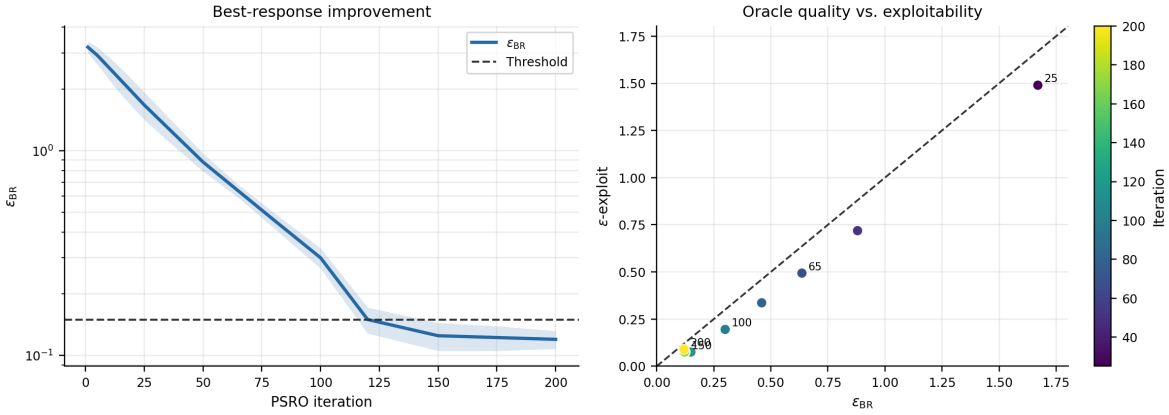

Figure 8: Best-response quality over PSRO iterations. Best-response improvement $\varepsilon_{\mathrm{BR}}$ decreases as the population matures, and empirical exploitability stays below the best-response proxy throughout, consistent with the convergence theorem's $\varepsilon_{\mathrm{BR}}$ slack term.

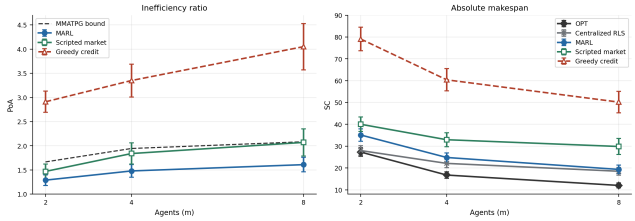

Figure 9: Scaling with agent count. PoA rises with m but remains below the bound; absolute makespan falls as additional agents provide parallelism.

Table 7: PNT benchmark statistics and Agora market settings.

Metric	Value
Total declarations	6,563
Sorry'd targets	92
Easy / Medium / Hard	39 / 41 / 12
Internal dependency edges	27,898
Dependency DAG max layer	37
Average internal dependencies	4.25
Initial cash per wallet	1,000
Run timeout	3,600 s
Bounty offer price (ℓ)	0.1
Offer units per target	100

provides parallelism but no explicit market or dependency-aware coordination. **A3: Aristotle** (Achim et al. 2025) and **A4: Open Gauss** (Math, Inc. 2026) are treated as black-box agentic theorem-proving baselines. **B1: Agora scaffold (8)** uses verified publication and bounty markets but no learned strategy. **C1: Agora + strategy (8)** adds the zero-shot strategy recommendations.

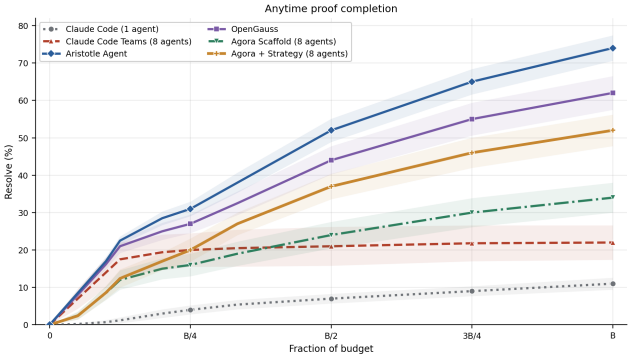

Figure 10: Anytime proof completion on the PNT benchmark. The learned strategy module improves Agora throughout the run, with gains appearing after dependency-unlocking publications.

Table 8 and Figure 10 report the main result. A single Claude Code agent solves 11% of targets. Naive parallelism (A2) doubles this to 22% but with high duplication and no dependency-awareness. The Agora scaffold (B1) reaches 34%, showing that verified publication and bounty markets already help even without learned coordination. **Adding the learned strategy module (C1) reaches 52%**, closing roughly half the gap to the purpose-built provers A3 and A4. Gains concentrate on Medium (24% $\rightarrow$ 41%) and Hard (8% $\rightarrow$ 25%) tiers, where dependency-aware sequencing and helper-lemma discovery matter most.

Table 9 reports coordination quality. Duplication falls

Table 8: Proof completion on the PNT benchmark by difficulty tier. Agora systems use 8 agents.

System	Easy	Medium	Hard	Overall
A1: Claude Code (1)	8 (21%)	2 (5%)	0 (0%)	10 (11%)
A2: Claude Code Teams (8)	15 (38%)	5 (12%)	0 (0%)	20 (22%)
A3: Aristotle Agent	35 (90%)	28 (68%)	5 (42%)	68 (74%)
A4: Open Gauss	32 (82%)	22 (54%)	3 (25%)	57 (62%)
B1: Agora scaffold (8)	20 (51%)	10 (24%)	1 (8%)	31 (34%)
C1: Agora + strategy (8)	**28 (72%)**	**17 (41%)**	**3 (25%)**	**48 (52%)**

Table 9: Coordination metrics for multi-agent PNT systems.

System	Duplication% ↓	DepUnlock ↑
A2: Claude Code Teams	36%	2
B1: Agora scaffold	21%	5
C1: Agora + strategy	**8%**	**9**

from 36% (no coordination beyond file locking) to 21% (shared library visibility plus market signals) to 8% (learned strategy). Dependency-unlock events grow from 2 to 9. The system still trails A3 and A4 in raw completion, which reflects prover quality at the LLM layer; the strategy module is a *coordination* layer, and the gap to those systems should be read as evidence for orthogonality: a stronger prover could be placed underneath the same market-mediated coordination layer and the gains compound.

The real-world gain over scaffold-only Agora (34% → 52%) is smaller than the synthetic gain (PoA from 2.91 to 1.29), likely due to the fact that the synthetic environment exactly matches the training game while the Lean deployment does not. In Lean, the proof kernel is a real LLM interaction loop with file edits, compiler feedback, tool failures, and mathematical content; the strategy module sees a compressed state while the LLM sees a rich proof context. That the strategy module nonetheless improves completion by 18 points over the scaffold suggests that the learned allocation heuristics do transfer through the distribution shift, even when the proof-generation layer is the bottleneck.

Coordination at the trace level. A representative example is the `SmallSquareInRectangle` chain in the PNT run: a Medium target whose proof unfolded across three agents and seven declarations, three of which were introduced mid-proof as conjectured helpers and propagated through the public library to unblock parallel branches. The full trace—dependency graph, timeline, and the actual Lean proofs accepted by VerifiedAgora—is in Appendix I (p. 18).

10 Conclusion

Large-scale theorem proving is not just proof search. It is a coordination problem over shared formal infrastructure. The MATPG formalism makes this explicit: agents have private libraries, uncertain proof capabilities, interdependent tasks, and individual incentives. Credit-based rewards create publication externalities and unbounded PoA via capability misallocation, so useful work goes to the wrong agents and helper lemmas may be withheld or delayed. A verifiable contract market changes the incentives by paying for public proof events, and the resulting MMATPG admits a $\frac{1}{1-\ell}(2-\frac{1}{m})$ PoA bound. Combined with PSRO+MAPPO, the learned policies publish, trade, and allocate proof effort in ways that improve both synthetic game outcomes and real Lean proof completion. Markets give the agents a reason to share; learning gives them a way to coordinate.

The broader implication is that future theorem-proving systems should separate two layers. The *prover layer* searches for proofs; the *coordination layer* decides what work is valuable, when to share it, and how to align incentives. The two layers have different literatures—neural theorem proving on one side, mechanism design and MARL on the other—and progress in either compounds with progress in the other.

10.1 Limitations

Synthetic-to-real gap. The synthetic proof kernel is stationary and Markovian. However, Real LLM agents are neither, as they learn within a conversation, change behavior after errors, and depend on prompt history. Zero-shot transfer is a practical choice and is not a substitute for fine-tuning on real-world Lean trajectories.

Restricted implementation. The implemented environment compresses sequents to formula targets under canonical contexts. The abstract theory allows arbitrary antecedents and multiple possible witnesses; this restriction is necessary for tractable training but omits proof-level richness.

Market simplifications. The deployed market supports target-conditioned binary contracts with simple deadlines and bounty seeding. Combinatorial contracts, automated market makers, or richer bounty design could improve liquidity and signaling but would increase strategic complexity.

Evaluation scope. The real-world evaluation is a single repository, a one-hour budget, and a limited set of systems. Having more comprehensive evaluations in the real-world would solidify the fact that market-mediated coordination improves an agent team on a dependency-heavy Lean benchmark.

Theory assumptions. The PoA guarantee assumes sufficient liquidity and positive gains from trade. These are

meaningful, since if no agent can prove a needed sequent, no market can force resolution. A natural direction, then, is to characterize efficiency under liquidity constraints and bounded rationality.

10.2 Future Work

Domain adaptation. The strategy module is currently trained on synthetic games and deployed zero-shot. A stronger system would collect real Agora trajectories and fine-tune on them, using which recommendations the LLM followed, which proof attempts succeeded, and which publications unlocked downstream targets as training signal. Imitation/warm-starting from expert trajectories of successful Lean agents is also natural.

Broader benchmarks and longer runs. PNT is dependency-heavy and mathematically meaningful, but a single repository. Future evaluation should include more MiniCTX projects, more mathematical domains, and longer time budgets. The market should help most on repositories with non-trivial dependency structure and may help less on collections of independent easy lemmas. Longer runs would also test whether markets create compounding advantages: in a one-hour budget only a few rounds of helper discovery and publication are possible, but in multi-day formalization, early infrastructure may unlock much larger downstream regions.

Richer mechanisms. The current contract space is intentionally simple. Combinatorial contracts that price bundles or prerequisite chains, automated market makers for continuous liquidity, or downstream-usage credit (rewarding lemmas proportional to subsequent reuse) are natural extensions; they may improve signaling but introduce new strategic behavior.

Asynchrony and resource models. The formal model uses discrete decision epochs and budgeted proof attempts. Real agents are asynchronous and heterogeneous. A continuous-time or token-budget model would better capture differing inference speeds and prompt costs, and would allow the market to price scarce compute directly.

Cooperative variants and human teams. Our focus on the non-cooperative regime reflects realistic deployment of heterogeneous agents and credit-seeking humans. A cooperative or mixed human/AI variant—closer to a Dec-POMDP (Bernstein et al. 2002)—would model collaborative formalization teams more directly.

References

Achim, T.; et al. 2025. Aristotle: IMO-Level Automated Theorem Proving. arXiv:2510.01346.

Ahuja, R. 2026. *Multi-Agent Theorem Proving*. Master's thesis, Carnegie Mellon University, Department of Mathematics.

Anthropic. 2025. Claude. https://www.anthropic.com/claude.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4): 819–840.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing*, 39(1): 195–259.

de Moura, L.; and Ullrich, S. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE 28*, 625–635. Springer.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

Graham, R. L. 1966. Bounds for Certain Multiprocessing Anomalies. *Bell System Technical Journal*, 45(9): 1563–1581.

Han, J. M.; Rute, J.; Wu, Y.; Ayers, E. W.; and Polu, S. 2022. Proof Artifact Co-training for Theorem Proving with Language Models. In *International Conference on Learning Representations*.

Hart, S.; and Mas-Colell, A. 2000. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica*, 68(5): 1127–1150.

Hu, J.; Zhu, T.; and Welleck, S. 2024. miniCTX: Neural Theorem Proving with (Long-)Contexts. arXiv:2408.03350.

Koutsoupias, E.; and Papadimitriou, C. H. 1999. Worst-Case Equilibria. In *STACS 1999*, 404–413. Springer.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Perolat, J.; Silver, D.; and Graepel, T. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 30.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30.

Math, Inc. 2026. Open Gauss: Project-Scoped Lean Workflow Orchestrator. https://github.com/math-inc/OpenGauss.

Myerson, R. B. 1981. Optimal Auction Design. *Mathematics of Operations Research*, 6(1): 58–73.

Nipkow, T.; Paulson, L. C.; and Wenzel, M. 2002. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer.

Pigou, A. C. 1920. *The Economics of Welfare*. Macmillan.

Polu, S.; and Sutskever, I. 2020. Generative Language Modeling for Automated Theorem Proving. arXiv:2009.03393.

Roughgarden, T.; and Tardos, É. 2002. How Bad Is Selfish Routing? *Journal of the ACM*, 49(2): 236–259.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms.

The Coq Development Team. 2024. The Coq Proof Assistant. https://coq.inria.fr. Version 8.19.

The mathlib Community. 2020. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, 367–381.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30.

Yang, K.; Swope, A. M.; Gu, A.; Chalamala, R.; Song, P.; Yu, S.; Godil, S.; Prenger, R. J.; and Anandkumar, A. 2023. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. In *Advances in Neural Information Processing Systems*, volume 36.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, volume 35, 24611–24624.

Zheng, K.; Han, J. M.; and Polu, S. 2021. MiniF2F: A Cross-System Benchmark for Formal Olympiad-Level Mathematics. arXiv:2109.00110.

A Full POSG and MMATPG Specification

This appendix gives the full POSG specification of the MATPG omitted from Section 4.

POSG. A POSG is a tuple $\mathcal{G} = (\mathcal{N}, \mathsf{S}, \{\mathcal{A}^\alpha\}, \{\mathsf{O}^\alpha\}, T, T^0, \{Z^\alpha\}, \{r^\alpha\}, \gamma)$, where $\mathcal{N}$ is a finite set of agents; S is the state space; $\mathcal{A}^\alpha, \mathsf{O}^\alpha$ are agent action/observation spaces; $T(\cdot|s,a)$ is the transition kernel on joint actions; T^0 is the initial distribution; $Z^\alpha(\cdot|s,a,s')$ is agent α's observation kernel; $r^\alpha : \mathsf{S} \times \mathcal{A} \to \mathbb{R}$; $\gamma \in [0,1)$ is the discount factor. Each agent's expected return is $R^\alpha(\pi) = \mathbb{E}^\pi[\sum_t \gamma^t r^\alpha(s_t, a_t)]$.

MATPG **state.** The MATPG state is

$$s_t = \big(\{\mathcal{L}_t^\alpha\}_\alpha, \mathcal{L}_t^0, \{J_t^\alpha\}_\alpha, \{Q_t^\alpha\}_\alpha, \{\Omega_t^\alpha\}_\alpha, \{U_t^\alpha\}_\alpha\big),$$

where J_t^α is α's active proof/conjecture jobs, Q_t^α is α's query model state, Ω_t^α is the (sequence of) query responses received by α, and U_t^α is α's cumulative effort map. The action space contains $\mathrm{Prove}, \mathrm{Conj}, \mathrm{Pub}, \mathrm{Qry}, \perp$ subject to admissibility constraints (proof actions only on local concrete sequents without active jobs, etc.). The observation kernel is deterministic: $Z^\alpha(o^\alpha|s,a,s') = \mathbf{1}[o^\alpha = \mathrm{Obs}^\alpha(s')]$ with $\mathrm{Obs}^\alpha(s) = (\mathcal{L}^\alpha, \Omega^\alpha, \mathcal{L}^0)$.

MMATPG **extension.** The world state extends s_t with a market state $\mathcal{M}_t = (\{\mathcal{P}_t^\alpha\}_\alpha, \mathcal{ML}_t)$, where $\mathcal{P}^\alpha = (c^\alpha, H^\alpha, O^\alpha)$ holds cash, positions, and posted offers and $\mathcal{ML}_t$ is the public order book. Action spaces are augmented with matched-pair issuance, post / accept / cancel offers; admissibility includes the full-collateralization constraint $\underline{B}^\alpha \geq 0$ where $\underline{B}^\alpha = c^\alpha - \sum_{x \in H^\alpha \cup O^\alpha} \sup_\omega(-\Pi_\omega(x))_+$. Observations gain $(\mathcal{P}_t^\alpha, \mathcal{ML}_t)$. Reward is $r^\alpha_{\mathrm{mkt}}(w_t, a_t) = B^\alpha_{t+1} - B^\alpha_t$. The transition is two-stage: non-market transitions (proof/conjecture/publication) first; then market settlement and trade.

B Full Action Kernels

Proof kernel. For agent α, $\mathcal{K}^\alpha_{\mathrm{proof}} : Y_{\mathrm{proof}} \times X_{\mathrm{proof}} \to [0,1]$ with $X_{\mathrm{proof}} = L \times \mathcal{S} \times \mathbb{N}^+$ (L the library-state space) and $Y_{\mathrm{proof}} = \{(0,\perp)\} \cup \{(1,d) : d \subseteq_{\mathrm{fin}} \mathcal{F}\}$. On admissible inputs $(\mathcal{L}, [\Gamma \vdash \phi], \tau)$ a successful output $(1,d)$ requires $d \subseteq \Gamma$.

Conjecture kernel. $\mathcal{K}^\alpha_{\mathrm{conj}}$ has the same input shape and output $\{(0,\perp)\} \cup \{(1,\psi) : \psi \in \mathcal{F}\}$. A successful output adds $\psi, \neg\psi$ to α's private concrete set.

Publication closure. For $R \subseteq \mathcal{RS}^\alpha_t$, define $\mathrm{cl}^\alpha_t(R) = \{s \in \mathcal{RS}^\alpha_t : \exists r \in R$ with directed path $s \to r$ in $D(\delta^\alpha_t)\}$. $\mathrm{Pub}^\alpha(C,R)$ is admissible only if $\mathrm{cl}^\alpha_t(R) \subseteq R$. The closure-completion is $\overline{(C,R)} = (\overline{C}, \overline{R})$ with $\overline{R} = \mathrm{cl}^\alpha_t(R)$ and $\overline{C} = C \cup \bigcup_{[\Gamma\vdash\phi]\in\overline{R}}(\Gamma \cup \{\phi\})$.

Query model. Each $Q^\alpha_t \in \mathcal{Q}$ is a stateful object with readout $q^\alpha : \mathcal{Q} \times \mathcal{S} \to [0,1] \times \mathbb{N}$ returning $(\hat{p}^\alpha_t(s), \hat{\tau}^\alpha_t(s))$ and an order-independent batch update operator $\Delta^\alpha : \mathcal{Q} \times (\mathcal{S} \times (\mathbb{N}^+ \times \mathcal{N})) \to \mathcal{Q}$.

C Centralized RLS

Let W be the total expected work and D the serial dependency depth. Any policy satisfies $\mathrm{OPT} \geq \max\{W/m, D\}$. RLS completes by time

$$C^{\mathrm{RLS}}_{\max} \leq \frac{W}{m} + \left(1 - \frac{1}{m}\right) D \leq \left(2 - \frac{1}{m}\right) \mathrm{OPT},$$

by an analogue of Graham's bound (Graham 1966). Dependencies revealed by proof attempts are added to the queue when they appear, so RLS is reactive: under perfect public information and the closure assumptions of Section 3.1, the same accounting applies in expectation.

D Externality and Unbounded PoA

Proof of Theorem 2 (mis-internalization). Let $\mathcal{N} = \{\alpha, \beta\}$ and let s be a sequent with $v_\beta(s) > 0$ and $v_\alpha(s) = 0$. Pick an instance and a state ω_t at which (i) $s \notin \mathcal{RS}^0_t$, (ii) α can resolve s at expected cost $c_\alpha(s) < \infty$, and (iii) β's set of feasible credited targets at time t is bounded away from optimum until $s \in \mathcal{RS}^0$. Consider the unilateral action sequence in which α proves and publishes s at time t. Let V^γ_t denote γ's value function under credit rewards. The action increases β's feasible-completion set, so V^β_t strictly increases (any positive-value target whose dependency s becomes feasible contributes a strictly positive expected discounted credit term to V^β that was zero before). The same action displaces $c_\alpha(s)$ steps of effort that α could otherwise have spent on its own credited targets, so V^α_t weakly decreases. This satisfies the externality definition. Since β's gain is independent of α's reward and no own-contribution function r^α depending only on α's actions and library state can mirror β's value function over arbitrary states, no own-contribution reward internalizes the cross-agent value of public resolution. □

Proof of Theorem 3 (capability misallocation). Fix $K > 0$ and instantiate the MATPG as follows. Two agents $\mathcal{N} = \{\alpha, \beta\}$ with proof kernels engineered so that the expected number of timesteps to resolve a target satisfies $c_\alpha(s_h) = 1$, $c_\beta(s_h) = K$, $c_\alpha(s_0) = K$, $c_\beta(s_0) = 1$; formally, choose Bernoulli proof rates $\rho^\alpha(s_h) = 1$, $\rho^\beta(s_h) = 1/K$, $\rho^\alpha(s_0) = 1/K$, $\rho^\beta(s_0) = 1$, and budget $\tau = 1$ throughout, so that $\mathbb{E}[\#\mathrm{steps}] = 1/\rho$. The latent dependency structure is $\delta^*(s_0) = \{\phi_h\}$ and $\delta^*(s_h) = \varnothing$. Credit values are $v_\beta(s_0) = 1$ and zero otherwise. The target set is $\Phi^* = \{s_0\}$.

Equilibrium social cost. Under credit rewards α's expected return is identically zero on every action sequence: α holds no target with positive credit and no sequent that depends on s_h has positive credit for α. Any best response of α is therefore action-payoff indifferent and we may select the convention that α does not attempt s_h (formally: among action profiles that are best responses we select the one in which α never spends effort on s_h; equivalently, perturb α's reward by an arbitrarily small idle bias, breaking ties toward the no-attempt action). Under this profile, β is the only agent who can make progress on s_0, and must first resolve s_h. The expected makespan is $\mathbb{E}[c_\beta(s_h)] + \mathbb{E}[c_\beta(s_0)] = K + 1$.

Centralized social cost. The centralized planner assigns s_h to α in parallel with β's wait. $\mathbb{E}[c_\alpha(s_h)] = 1$ unit of

time makes s_h public; β then resolves s_0 in expected time $\mathbb{E}[c_\beta(s_0)] = 1$. Total $\mathrm{OPT} = 2$.

Ratio. $\mathrm{PoA} \geq (K+1)/2$. Letting $K \to \infty$ gives unbounded PoA.

Tie-breaking robustness. The argument is robust to tie-breaking: if α does attempt s_h with arbitrary positive probability $\eta > 0$ but no contract reward, the equilibrium social cost is at least $(1-\eta)(K+1)+\eta(1+1) = K+1-\eta(K-1)$, which still scales with K for any $\eta < 1 - O(1/K)$. The bound becomes vacuous only if α behaves as if fully aligned with β, which is not implied by credit rewards.

Why a market closes the gap. Consider the same instance under the MMATPG. β issues a matched pair on the contract type $\chi = (s_h, T_1, \ell)$ at any unit price $p \in (0,\, 1-\ell-1/q\,]$ for q sufficiently large, retaining the short side and selling the long to α. α accepts at total premium qp, proves s_h at expected cost 1, and collects $q(1-\ell)$ on settlement, with net payoff $q(1-\ell-p)-1 > 0$. β's side of the trade is individually rational for K sufficiently large: paying $q(1-\ell-p)$ on settlement (after netting premium) saves β the K-step cost of self-proving s_h, and β can then prove the credited target s_0 at expected cost 1. The market thereby restores the centralized parallel allocation and the makespan returns to $\mathrm{OPT} = 2$.

Remark (alternative failure mode: intermediate hoarding). Theorem 3 can also be proven via a hoarding construction. Take agents $\alpha_0, \ldots, \alpha_n$ and target set $\Phi^* = \{s_0\}$ with chain $\delta^*(s_0) = \{\phi_1\}, \ldots, \delta^*(s_{n-1}) = \{\phi_n\}$. Each α_i has a private credited target s_i' that depends on s_i, and resolves s_i as a byproduct. If $v_\alpha(s_i) = 0$ for every agent other than one with no downstream credit dependency on its earlier publication, no agent has strict incentive to publish s_i, so s_0 never resolves within the horizon. This construction illustrates the second consequence of Theorem 2 but relies on a more delicate reward-and-tie-breaking specification than the heterogeneous-cost construction above. The capability-misallocation construction is the headline because it survives plausible perturbations of private credited targets, idle biases, and partial information. □

E Market Internalization and Bounded PoA

Proof of Theorem 5. Suppose β buys q units of long contracts $\chi^+ = (s, T, \ell)$ at price $p < v_\beta$. If $E_{s,T} = 1$, settlement pays β a net $q(1-\ell-p)$; if $E_{s,T} = 0$, β loses $q(\ell+p)$. As long as β's subjective probability of resolution exceeds $(\ell+p)/(1-p+\ell)$, the trade has positive expected surplus.

For agent α, acquire q units of χ^+ at price p (purchase or matched issue and short-sale of the complementary side). α resolves s at cost c_α and publishes, triggering $E_{s,T} = 1$. Net payoff $q(1-\ell) - qp - c_\alpha$. For any $q \geq c_\alpha/(1-\ell-p)$ this is nonnegative; in particular, for $p < 1-\ell-c_\alpha/q$ the trade is strictly profitable.

Settlement reads $\mathcal{RS}_T^0$, not $\mathcal{RS}_T^\alpha$. An agent who proves s privately but does not publish has $s \in \mathcal{RS}_T^\alpha$ but $s \notin \mathcal{RS}_T^0$, so $E_{s,T} = 0$ and the agent does not collect. □

Proof of Theorem 6. Step 1: every required sequent resolves. Suppose for contradiction that some required $s \in \Phi^* \cup \delta^*(\Phi^*)$ is unresolved at NE. By assumption (iii), there is β with $v_\beta(s) > c_\alpha(s)$ for some prover α. Consider the unilateral deviation: any agent γ issues a matched pair (χ^+, χ^-) for (s, T, ℓ) at large T, sells χ^- to β at price $p^- > 0$ (which is individually rational for β if β believes s will not be resolved, or buys the long side if β believes it will), proves and publishes s, and collects the long settlement. The deviation is profitable when $1-\ell+p^- > c_\gamma(s)$, which holds under sufficient liquidity. This contradicts NE.

Step 2: makespan bound. Once publication is incentivized, the market prices coordinate agents toward positive-surplus tasks and the balance-change reward favors prompt resolution (later resolution means more discounting). In the worst case, agents behave like a reactive list scheduler in expectation. Following the RLS argument (Appendix C), the resulting makespan satisfies

$$\mathrm{SC}(\pi^*_{\mathrm{NE}}) \leq \frac{1}{1-\ell}\left(\frac{W}{m} + \left(1-\frac{1}{m}\right)D\right) \leq \frac{1}{1-\ell}\left(2-\frac{1}{m}\right)\mathrm{OPT},$$

where the $1/(1-\ell)$ accounts for the loss-parameter friction: agents require a positive profit margin to cover ℓ, which in the worst case inflates the schedule by a factor of $1/(1-\ell)$. □

CCE extension. Let μ be a policy-space CCE of the MMATPG. Suppose a required sequent s is left unpublished with positive probability under μ. The deviation in Step 1 above is unconditional: it is profitable against any opponent mixture because the settlement payoff depends only on the deviating agent's own publication action. So the deviation violates the CCE inequality, a contradiction. Hence the same scheduling bound applies. For an ε-CCE the deviation may have surplus at most ε, so sequents with resolution surplus below ε may persist; this adds an $O(\varepsilon H/(1-\ell))$ term to the makespan bound, where H is the horizon.

F MARL Convergence

F.1 Policy-space CCE for POSGs

A distribution μ over joint policies is a policy-space CCE for a POSG if for every α and $\hat{\pi}^\alpha$, $\mathbb{E}_{\pi\sim\mu}[R^\alpha(\pi)] \geq \mathbb{E}_{\pi^{-\alpha}\sim\mu^{-\alpha}}[R^\alpha(\hat{\pi}^\alpha, \pi^{-\alpha})]$. An ε-CCE relaxes the inequality by ε on the right.

F.2 No-regret meta-strategy ⇒ meta-game CCE

If meta-strategy selection in PSRO is a no-regret algorithm (multiplicative weights, regret matching, etc.), then by the standard no-regret-to-CCE theorem (Hart and Mas-Colell 2000), after T iterations the empirical meta-strategy distribution is a $\mathrm{Reg}(T)/T$-CCE of the meta-game. With $\mathrm{Reg}(T) = O(\sqrt{T \log K})$ and $K \leq T$, this is $O(\sqrt{\log T/T})$.

F.3 Meta-game CCE ⇒ full-game ε-CCE

Let the best-response approximation error of the populations be $\varepsilon_{\mathrm{BR}}(\sigma) = \max_\alpha[\sup_{\pi^\alpha} R^\alpha(\pi^\alpha, \sigma^{-\alpha}) - \max_{\pi^\alpha \in \mathcal{P}^\alpha} R^\alpha(\pi^\alpha, \sigma^{-\alpha})]$. If μ^{meta} is an $\varepsilon_{\mathrm{meta}}$-CCE of the meta-game and $\varepsilon_{\mathrm{BR}} \leq \varepsilon$ on the support of μ^{meta}, the induced joint-policy distribution $\mu = \mu^{\mathrm{meta}} \circ (\pi_\cdot^\alpha)_\alpha$ is an

$(\varepsilon_{\text{meta}} + \varepsilon_{\text{BR}})$-CCE of the full POSG: combining the meta-game inequality with the population approximation yields the full-game inequality up to that additive slack.

F.4 Main convergence theorem

Combining the previous two subsections, after T PSRO iterations the induced joint-policy distribution is an $\varepsilon(T)$-CCE of the full MMATPG with

$$\varepsilon(T) \le C\sqrt{\frac{\log T}{T}} + \varepsilon_{\text{BR}},$$

C depending on horizon and reward magnitude. As $T \to \infty$ and $\varepsilon_{\text{BR}} \to 0$, μ converges to the exact-CCE set. Combined with the ε-CCE PoA bound from Appendix E, the joint-policy distribution achieves

$$\frac{\mathbb{E}_{\pi\sim\mu}[\text{SC}(\pi)]}{\text{OPT}} \le \frac{1}{1-\ell}\left(2 - \frac{1}{m}\right) + O\Big(\tfrac{H}{(1-\ell)\text{OPT}}\left(\sqrt{\log T/T} + \varepsilon_{\text{BR}}\right)\Big).$$

G Implementation Details

Environment state. The implementation state is private libraries, the public library, query models, active proof / conjecture jobs, cumulative proof and conjecture effort maps, market state, and a timestep. A proof job records type, target formula, remaining duration, effective duration, and prior accumulated effort. Budget-one jobs can complete in the same step they are started.

Formula-level compression. The general theory treats sequents $[\Gamma \vdash \phi]$ explicitly. The implementation collapses sequents to formula targets under the canonical local context $\Gamma = \mathcal{R}_t^\alpha$, the agent's currently resolved formulas. This makes the action space linear rather than exponential. Under this restriction, every resolved formula is automatically fully resolved.

Synthetic kernel parameters. For agent α and formula ϕ, the rate is

$$\rho^\alpha(\phi) = e^{-\lambda_{\text{diff}} d(\phi)}\big(\rho_0^\alpha + \rho_1^\alpha \log(1 + \sum_\psi w(\psi,\phi)^\alpha)\big),$$

combining difficulty modulation with utility-weighted support. The diminishing-returns exponent $\kappa \in (0,1]$ controls the conditional Weibull. The conjecture kernel uses the same Weibull shape with a rate driven by local opportunity mass over ghost formulas anchored at the target. The query model is an oracle with persistent per-formula noise on both probability and time estimates.

Curriculum. Five difficulty levels with $n \in \{4, 8, 16, 32, 64\}$ theorem pairs; transitions triggered by exploitability dropping below a threshold on the current level.

Synthetic graph generator. Layered, clustered DAG generator parameterized by layer count, cluster count, self-affinity γ_{self}, profile concentration c_{profile}, foundation nodes $(n_{\text{found}}, p_{\text{found}})$, and a clipped-exponential in-degree distribution $([\tau_{\min}, \tau_{\max}], \mu_{\deg})$. Edges go only from higher to lower layers (acyclic by construction). Cluster-level Dirichlet dependency profiles produce the cross-cluster sparsity observed in real codebases. Per-node rarity is computed from the generated DAG with the same formula used for the real repository, then quantile-matched against a Mathlib-calibrated table and mapped to $[0.05, 0.95]$ difficulty.

Training stack. Actor / critic optimized with PPO-style clipped objective (Schulman et al. 2017), GAE advantages (Schulman et al. 2015), entropy regularization, AdamW (Loshchilov and Hutter 2019). CTDE: actor sees only o_t^α; critic receives privileged aggregate state. Population checkpoints added periodically. Opponent sampling at each rollout uses a mixture of self-play / population / scripted opponents. Code organization follows the conceptual split: `envs/vamp` for environment dynamics and market mechanics; `models/gpt_model.py` for actor/critic; `run_madt.py` for PPO rollouts and optimization; `run_curriculum.py` for the level transitions.

H Agora Architecture Details

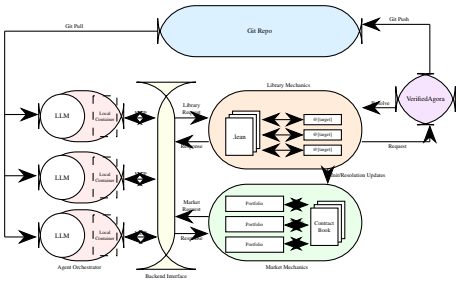

Figure 11: Agora platform architecture. Library Mechanics owns Lean validation; Market Mechanics owns financial invariants; the Backend Interface exposes MCP tools to LLM agents. The Agent Orchestrator provisions containers and seeds the market.

Library Mechanics. Owns Lean interaction. Maintains an LRU cache of project repositories on Agora-specific git branches; manages initialization, building, target extraction, and verification. A dependency-aware request scheduler serializes concurrent contributions to the same project, ensuring downstream module rebuilds are correct.

Market Mechanics. Owns wallets (cash, private assets, posted offers), target statuses, and the public order book. Enforces money conservation (zero net coefficient on each transaction, after simplification against current resolution data) and the liquidity invariant (each wallet's worst-case marked balance is nonnegative). On target resolution it records solve time and proving agent, simplifies referencing assets and offers, and converts determined payoffs into wallet balances.

Backend Interface. Public API gateway: agent authentication, target metadata sync, WebSocket activity broadcasts, and the MCP server exposed to LLM agents. Delegates Lean validation to Library Mechanics and financial invariants to Market Mechanics.

Agent Orchestrator. Claude Agent SDK-based harness provisioning agent containers and managing their lifecycles. Each container's LLM has MCP access to library tools (list projects, submit contributions), market tools (create / take / cancel offers, transfer assets), Lean LSP tools (goal state, hover info), and filesystem tools. The harness maintains structured agent state (active project, target, wallet, build status, market positions), with recovery hints for common failure modes (edit mismatches, compilation errors, path issues). Auto-compaction summarizes long conversations into a memory file.

VerifiedAgora. Lean library providing the trusted verification layer. The `@[target]` attribute marks proof obligations. Non-target declarations may transitively depend only on `propext`, `Quot.sound`, and `Classical.choice`; targets may additionally depend on `sorryAx` while unresolved, but a *solved* target must be sorry-free and axiom-clean. Validation: compile the submission, extract declarations, check type / kind / attribute / instance compatibility and axiom policy; produce an automerged file (valid solved targets replace sorry'd versions, invalid revert, new non-target declarations included), write it, and run `lake build`; if the build fails, restore the original. This pipeline ensures agents cannot break the repository.

Time discretization at inference. Agents are asynchronous in continuous wall-clock time, but the MMATPG is discrete-time with simultaneous actions. We discretize into decision epochs each corresponding to one LLM interaction step. Within an epoch, an agent takes one high-level action; proof attempts of budget τ occupy the strategic agent for τ epochs while the prover subagent iterates; publication and market actions are instantaneous. Budget therefore measures LLM interaction steps, not wall-clock seconds, normalizing across agents with different inference speeds.

I Qualitative Case Study: the `SmallSquareInRectangle` Chain

To illustrate how the scaffold distributes and sequences work across agents, we trace one dependency chain in the PNT run. The terminal target is `SmallSquareInRectangle`, a medium-difficulty lemma stating that if a complex point lies in the interior of a rectangle, all sufficiently small centered squares are contained in the rectangle. At the start of the run, *both* of its parent targets were sorry'd, and three of the seven declarations in the eventual chain did not exist as declarations at all: they were introduced as conjectures by agents during proof construction.

Bottom of the chain: conjectured helpers. Mid-attempt on `rect_subset_iff`, agent α_2 issued a Conj action whose proposal distribution placed mass on `left_mem_rect` and `right_mem_rect`: the corner points of a rectangle lie in it. Both helpers resolved in the same epoch ($\approx$ 1500 s) and were committed to the shared branch. After the publication broadcast, all other agents saw them on next reengagement.

First target. Equipped with the helpers, α_2 resolved `rect_subset_iff` itself ($\approx$ 2100 s). The forward direction of the iff used `left/right_mem_rect` directly; the reverse direction was discharged by `grind` after unfolding definitions.

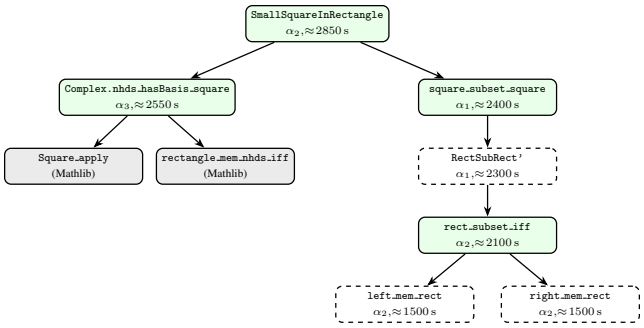

Figure 12: Dependency chain resolved by three agents. Solid green: original benchmark targets. Grey: pre-existing Mathlib lemmas. Dashed: conjectured helper lemmas introduced mid-proof, not in the original target set.

Parallel branches. α_3 was simultaneously proving the harder analytic branch `Complex.nhds_hasBasis_square` (the longest proof in the chain), whose upstream dependencies were already in Mathlib. It published at $\approx$ 2550 s. Meanwhile α_1, assigned `square_subset_square` by the strategy module, conjectured `RectSubRect'`—a stronger reformulation expressing rectangle inclusion in terms of componentwise inequalities—then resolved `square_subset_square` as a four-line invocation of `RectSubRect'` ($\approx$ 2400 s). The publication of `rect_subset_iff` immediately unblocked α_1's stalled proof; the publication of `RectSubRect'` immediately unblocked `square_subset_square`.

Closing the target. With both direct parents public, α_2 closed `SmallSquareInRectangle` at $\approx$ 2850 s.

Table 10: Timeline for the `SmallSquareInRectangle` chain.

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

Three features of this trace illustrate the coordination mechanisms at work. *Parallelism on independent subchains*: α_3 worked on the hardest proof concurrently with α_1, α_2 on the other branch; the subchains share only the final target, so no publications propagated between them until the last step. *Conjecture as task decomposition*: three of seven declarations did not appear in the original target set; they were introduced mid-proof as auxiliary abstractions. *Publication-triggered cascades*: each publication unblocked a stalled downstream attempt, with corresponding reweighting of open market offers.

Listings. The proofs accepted by the run for this chain are reproduced below.

```
-- left_mem_rect, right_mem_rect (alpha_2, ~1500s)
lemma left_mem_rect (z w : C) : z in Rectangle z w := by
  apply And.intro (by norm_num) (by norm_num)

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

-- rect_subset_iff (alpha_2, ~2100s)
lemma rect_subset_iff {z w z' w' : C} :
    Rectangle z' w' subset Rectangle z w iff
    z' in Rectangle z w and w' in 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

-- RectSubRect' (alpha_1, ~2300s)
lemma RectSubRect' {z0 z1 z2 z3 : C} (...) :
    Rectangle z1 z2 subset 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
          ) ]

-- square_subset_square (alpha_1, ~2400s)
lemma square_subset_square {p : C} {c1 c2 : R}
    (hc1 : 0 < c1) (hc : c1 le c2) :
    Square p c1 subset Square p c2 := by
  apply RectSubRect';
  all_goals norm_num [ Complex.ext_iff ] at *; linarith

-- Complex.nhds_hasBasis_square (alpha_3, ~2550s; longest
    proof)
theorem Complex.nhds_hasBasis_square (p : C) :
    (nhds p).HasBasis (0 < .) (Square p .) := by
  refine' < fun s => < fun hs => _, fun hs => _ > >;
  . rw [ Metric.mem_nhds_iff ] at hs;
    have h_square_ball : forall e > 0,
        Square p (e/2) subset Metric.ball p e := by
      intro e he;
      intro x hx; rw [ Square_apply p ( half_pos he ) ] at hx;
      simp_all +decide [ Complex.dist_eq, Complex.normSq ];
      exact Real.sqrt_lt' he |>.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 > >

-- SmallSquareInRectangle (alpha_2, ~2850s; final target)
lemma SmallSquareInRectangle {z w p : C}
    (pInRectInterior : Rectangle z w in nhds p) :
    forall_eventually (c : R) in nhds_within > 0,
        Square p c subset Rectangle z w := by
  obtain < e, he_pos, he > : exists e > 0, forall c : R,
      0 < c -> c < e -> Square p c subset Rectangle z w := by
    obtain < e, he > : exists e > 0,
        Square p e subset z.Rectangle w := by
      have := Complex.nhds_hasBasis_square p;
      exact this.mem_iff.mp pInRectInterior;
    exact < e, he.1, fun c hc1 hc2 =>
      Set.Subset.trans
        ( square_subset_square ( by linarith ) ( by linarith )
            )
        he.2 >;
  filter_upwards [ Ioo_mem_nhdsGT he_pos ] with c hc using
    he c hc.1 hc.2
```

J Reproducibility Notes

The synthetic figures in the main paper are generated from saved evaluation artifacts rather than hand-drawn. Some original raw data and larger training outputs are not present in the current repository snapshot, so the paper reports the available generated artifacts and tables rather than claiming a fully rerunnable end-to-end pipeline. The paper specifies the environment specialization, kernel form, market mechanics, baselines, and reported metrics; a fully reproducible release would additionally require all random seeds, raw rollout logs, Lean agent transcripts, and trained checkpoints used for the PNT deployment.

For the Lean run, duplicated work is counted at the target level rather than at the tactic-state level: two agents pursuing different helper lemmas for the same final target are not counted as duplicates unless they both attempt the same benchmark declaration after it is solved. Dependency unlocks are counted when a public contribution changes another target's hard-feasibility status—a conservative metric that excludes soft-support improvements.