INEQAGENT: AGENTIC INEQUALITY PROVING IN LEAN WITH SYMBOLIC COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Formal inequality proving requires mathematical reasoning and exact computation, often connected through intermediate lemmas and goal reformulations. We introduce IneqAgent, an agentic framework combining LLM reasoning with symbolic computation for inequality proving in Lean 4. Lemma-level Monte Carlo Tree Search coordinates lemma construction, goal reformulation, and symbolic backend selection. Complementary sum-of-squares, Bernstein, and successive difference-substitution backends produce exact certificates for local Lean proof generation. The agent integrates these proofs with LLM-generated steps and uses Lean feedback to construct complete proofs checked against the original theorem statements. On four benchmarks, IneqAgent with DeepSeek-V4-Flash solves 178 of 190 problems (93.68%), achieving the highest overall success rate among the evaluated methods. Across four backbones, it achieves higher success rates at low LLM inference cost thresholds than Direct Agent using the same backbone, with lower mean inference costs over each method's successful runs. We also obtain Lean-verified strengthenings of Dittert's conjecture in dimensions four and five, and certify the Holens–Djokovic $(4, 4)$ inequality, implying Merris's conjecture in dimension four.

1 INTRODUCTION

Inequality proving is an important task in automated mathematical reasoning (Wei et al., 2024; Li et al., 2025a). Even concise inequalities can require carefully chosen substitutions, auxiliary bounds, and substantial algebraic computation. Proving them in Lean (de Moura & Ullrich, 2021) requires suitable mathematical strategies and explicit justification of each step, including sign and domain conditions.

Large language models (LLMs) support formal proof construction through learned proof generation and interaction with tools and verifiers (Lin et al., 2025; Ren et al., 2025; Wang et al., 2025; Varambally et al., 2026; Requena et al., 2026). Symbolic methods complement these capabilities by constructing checkable algebraic certificates (Peyrl & Parrilo, 2008; Kaltofen et al., 2012; Magron & Safey El Din, 2018). Combining them presents two challenges. First, symbolic backends depend on the goal's algebraic form and assumptions: a rational inequality, for example, may require denominator positivity before polynomial reformulation. Second, exact certificate construction can be computationally demanding. When numerical solvers are used, floating-point candidates typically satisfy certificate identities only approximately and require exact recovery and verification.

We introduce IneqAgent, an agentic framework combining LLM reasoning, symbolic computation, and Lean verification. The agent uses lemma-level MCTS to explore proof paths, choosing backend calls, goal reformulations, and intermediate lemmas according to the goal's structure and backend requirements. It integrates certificate-derived local proofs with LLM-generated steps, rewrites, and proofs of side conditions. Lean checks the complete proof against the original theorem statement.

The symbolic module provides three complementary backends. Using Putinar-type certificates (Putinar, 1993), the sum-of-squares (SOS) backend extends the numerical–symbolic certification procedure of Kaltofen et al. (2012) from unconstrained problems to problems with equality and inequality constraints through sequential recovery. The Bernstein backend combines numerical and exact linear programming with subdivision for box-constrained goals. Successive difference

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

substitution uses exact substitutions and coefficient nonnegativity checks for polynomial goals over nonnegative variables. The module supports applicable original goals and intermediate lemmas.

We apply IneqAgent to conjectures on matrix permanents, obtaining Lean-verified proofs of strengthened inequalities for Dittert's conjecture in dimensions four and five that establish quantitative stability and unique maximizers. We also verify an exact SOS certificate for the Holens–Djokovic $(4, 4)$ inequality in Lean and derive Merris's conjecture in dimension four. On four inequality benchmarks, IneqAgent with DeepSeek-V4-Flash solves 178 of 190 problems (93.68%), achieving the highest overall success rate among the evaluated methods. It also records lower mean solving times than the evaluated LLM-based provers. Across four backbones, it achieves lower mean LLM inference costs and higher success rates at low cost thresholds than Direct Agent using the same backbone. Both means are computed over each method's own successful runs. Ablations further support combining MCTS with symbolic computation.

Our main contributions are as follows:

- We introduce IneqAgent, which uses lemma-level MCTS to coordinate LLM reasoning, goal reformulation, and symbolic computation, producing complete proofs checked by Lean against the original theorem statements.
- We develop a symbolic proving module with complementary SOS, Bernstein, and difference-substitution backends. We extend the recovery procedure of Kaltofen et al. (2012) to constrained SOS certificates through sequential recovery, and translate certificates from all three backends into local Lean proofs.
- We demonstrate applications to matrix-permanent conjectures and introduce LitIneq, a benchmark of 78 formalized inequality tasks from research papers and formal-verification projects. We evaluate IneqAgent through benchmark comparisons, inference-cost analysis, and ablation studies.

2 Related Work

Symbolic Methods for Inequality Proving. General-purpose systems provide arithmetic and algebraic reasoning: Z3 supports SMT-based arithmetic solving (de Moura & Bjørner, 2008), while Maple provides symbolic manipulation and quantifier elimination (Heck, 1993). Symbolic-numeric computation methods convert approximate solutions into exact certificates (Peyrl & Parrilo, 2008; Kaltofen et al., 2012; Magron & Safey El Din, 2018). Other approaches exploit domain structure, including Bernstein representations with subdivision for box-constrained problems (Ray & Nataraj, 2009) and successive difference substitution for nonnegative variables (Xia & Yang, 2016). Isabelle/HOL's `Sum_of_Squares` library integrates SOS certificates with formal proof checking (Chaieb & Meyer, 2025).

LLMs for Formal Theorem Proving. Neural provers generate tactics from formal proof states and search over their continuations (Polu & Sutskever, 2020; Lample et al., 2022), with retrieval supplying relevant library lemmas (Yang et al., 2023). More recent models incorporate proof data synthesis, self-correction, and subgoal decomposition (Lin et al., 2025; Ren et al., 2025). Agentic systems combine these capabilities with tool use and verifier feedback. Hilbert recursively combines informal reasoning with formal subgoal proving (Varambally et al., 2026), while AxProverBase uses a modular loop of proof proposals, review, memory, and tool use (Requena et al., 2026).

LLMs with Symbolic Reasoning for Inequality Proving. Existing systems assign different roles to learned reasoning and symbolic computation. LIPS combines symbolic generation and pruning of scaling steps with LLM-generated rewrites and goal ranking, checking the resulting proofs in Lean (Li et al., 2025a). IneqSearch combines symbolic decomposition with an LLM agent that proposes transformations, and expands its theorem database with newly proved inequalities (Li et al., 2025b). For unconstrained polynomial inequalities, NSPI uses an LLM to conjecture approximate SOS decompositions, followed by numerical refinement, rational recovery, and Lean proof generation (Zuo et al., 2026). IneqAgent uses lemma-level MCTS to coordinate intermediate lemma construction, goal reformulation, and symbolic backend selection according to the current assumptions and algebraic structure. Its complementary backends produce exact certificates that are translated into local Lean proofs and combined with agent-generated steps to prove the original theorem.

3 METHODOLOGY

Problem formulation. We study the problem of proving inequalities in real variables. The expressions involved are constructed from real variables and constants using arithmetic operations and elementary functions, such as square roots, exponentials, logarithms, and trigonometric functions. For real variables $\mathbf{x} = (x_1, \ldots, x_n)$, a typical inequality-proving problem is to establish

$$f(\mathbf{x}) \geq 0 \qquad \text{for all } \mathbf{x} \in D, \tag{1}$$

where the domain D is specified by the assumptions of the theorem:

$$D := \{\mathbf{x} \in \mathbb{R}^n \mid q_i(\mathbf{x}) \geq 0,\ i = 1, \ldots, m;\ r_j(\mathbf{x}) = 0,\ j = 1, \ldots, \ell\}.$$

Here, f, q_i, and r_j are real-valued expressions of the form described above. When $m = \ell = 0$, we have $D = \mathbb{R}^n$, corresponding to an unconstrained problem. Strict inequalities are also allowed in both the constraints and the target. The input is a theorem statement formalized in Lean 4, and the task is to construct a complete proof accepted by the Lean kernel.

To address these problems, we propose IneqAgent, an agentic framework for automated inequality proving in Lean 4. The agent combines mathematical reasoning with symbolic computation to reformulate goals, construct intermediate lemmas, and assemble formal proofs. Computational feedback and Lean diagnostics guide its subsequent proving steps. The overall design is illustrated in Fig. 1. We describe the framework in Section 3.1 and the symbolic computation module in Section 3.2.

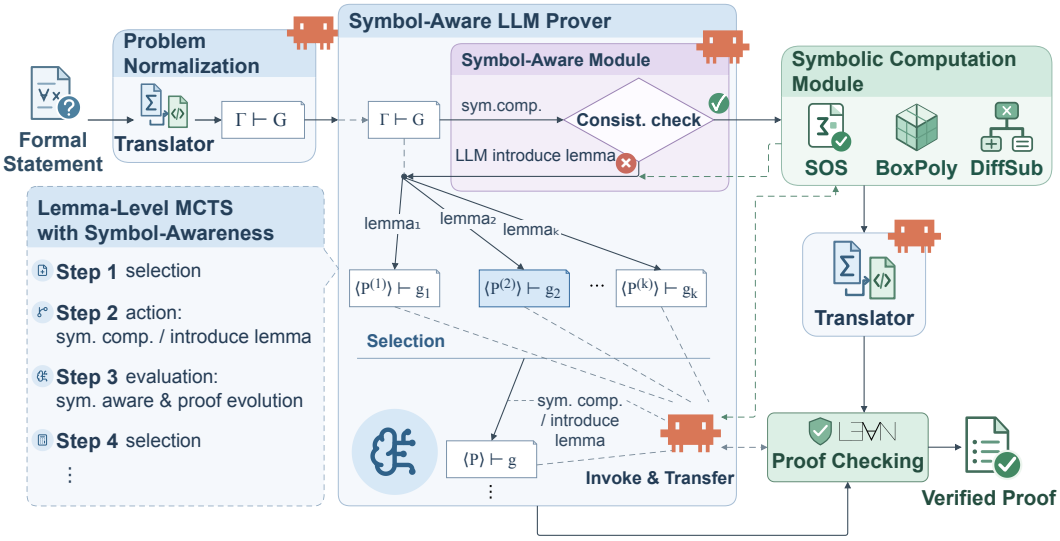


Figure 1: Overview of IneqAgent. The agent normalizes the formal statement and assesses each goal for symbolic computation. It develops intermediate lemmas when further reasoning is needed, coordinates tool calls, and integrates local proofs into the current proof context. Symbolic feedback and Lean diagnostics guide subsequent attempts until the original theorem is proved.

3.1 AGENTIC FRAMEWORK FOR INEQUALITY PROVING

The agent normalizes the statement to a nonnegativity target under inequality and equality constraints, retaining strict signs where needed. Expressions need not yet be polynomial, and subsequent transformations must preserve their side conditions. The normalized goal initializes lemma-level MCTS, which iterates through selection, action, and evaluation (Figure 1).

Selection: lemma-level MCTS. Each node represents a partial proof state containing the context, established lemmas, and remaining goals. An action attempts to extend this state through symbolic computation or by introducing and proving lemmas. Verified extensions create child states with fully proved local lemmas and updated goals. Each path records successive lemma-level extensions, while sibling nodes retain alternative continuations of the same proof state.

For each node v, $N(v)$ counts attempts backed up through it, $W(v)$ accumulates their rewards, and $Q(v) = W(v)/N(v)$ is their mean, initialized to zero for unvisited nodes. Thus, $Q(v)$ summarizes the outcomes of attempts through that branch. Selection starts at the root. Progressive widening permits new children according to visit counts and stagnation; otherwise, selection descends to the open child u maximizing

$$U(u \mid v) = Q(u) + c\sqrt{\frac{\log(N(v)+1)}{N(u)+1}} - p(u),$$

where c weights exploration of less-visited branches and $p(u)$ is a bounded penalty for repeated unproductive expansions. Selection continues until a state can be expanded. If no open child is available, a new extension is attempted at the current node. Widening and penalty rules are given in Appendix A.

Action: symbolic-aware lemma reasoning. The agent receives the selected state's context, goals, proof history, and tool feedback. It first assesses whether the goal's algebraic structure and constraints match a symbolic backend. Before invoking one, it checks that the structured variables, constraints, and target correspond to the Lean goal, including conditions required by any reformulation. This consistency check connects the selected proof state to the symbolic input.

If direct symbolic computation is unsuitable or unsuccessful, the agent uses the backend requirements and available assumptions to propose lemmas that support further reasoning or yield a backend-compatible goal. For a rational inequality, it may establish denominator positivity before reducing the target to polynomial nonnegativity. The agent attempts to prove each lemma through symbolic computation or LLM-generated Lean reasoning. Different lemma choices produce alternative proof extensions, each with its own established facts and remaining goals.

When symbolic computation succeeds, the agent invokes the Translator to convert the resulting certificate into a local Lean proof, then integrates it with the necessary rewrites and side conditions. The resulting candidate extension is submitted to Lean for checking. Unsuccessful symbolic attempts provide feedback for revising the input, trying another backend, or proposing different lemmas. Section 3.2 describes the symbolic computations.

Evaluation: Lean feedback and proof evolution. Lean feedback determines whether the extension completes the theorem, establishes valid lemmas with goals remaining, or fails verification. Failures return diagnostics for repair. For a valid open state, a Lean-based score measures verified progress, including changes in the remaining goals. Proof evolution is assessed by an LLM comparing the parent and candidate states to judge whether the established lemmas simplify the remaining proof and bring the theorem closer to completion. Both scores lie in $[0, 1]$ and are combined as $r = \lambda r^{\text{Lean}} + (1-\lambda) r^{\text{LLM}}$, where $\lambda \in [0, 1]$ weights Lean-based progress. Completed proofs receive reward one and failed verifications receive zero.

The same reward is backed up along the selected path: each visited node increments N by one, adds the reward to W, and recomputes Q. The next iteration starts at the root using these updated statistics; only valid open states can be selected. Search returns a complete Lean-verified proof without `sorry` or `admit`, or stops when the budget is exhausted or no open state remains. The reward formula and update rules appear in Appendix A.

3.2 Lean-Verified Symbolic Proving

The symbolic module addresses polynomial nonnegativity goals within the general problem formulation above. These goals are prepared by the agent and may coincide with the input theorem or arise from reformulations and intermediate lemmas during proof search. Specifically, the module seeks to establish

$$p(\mathbf{x}) \geq 0 \qquad \text{for all } \mathbf{x} \in K, \tag{2}$$

where

$$K := \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \geq 0,\ i = 1, \ldots, m;\ h_j(\mathbf{x}) = 0,\ j = 1, \ldots, \ell\},$$

and p, g_i, h_j are polynomials. The unconstrained case corresponds to $m = \ell = 0$. On success, the module returns an exact certificate for subsequent Lean proof generation and verification. If no certificate is obtained, it returns feedback to the agent for further proof search.

SOS-based proving under polynomial constraints. Motivated by Putinar's Positivstellensatz (Putinar, 1993), the SOS backend searches for a certificate of the form

$$p = \sigma_0 + \sum_{i=1}^{m} g_i \sigma_i + \sum_{j=1}^{\ell} h_j q_j, \tag{3}$$

where $\sigma_0, \ldots, \sigma_m$ are sums of squares and $q_1, \ldots, q_\ell$ are polynomial multipliers unrestricted in sign. Such an identity is sufficient to establish the nonnegativity of p on K. Imposing degree bounds on the SOS polynomials and equality multipliers yields a finite-dimensional SOS relaxation. Each SOS polynomial is represented as $\sigma_i = z_i^{\mathsf{T}} G_i z_i$, where z_i is a monomial vector and $G_i \succeq 0$ is a Gram matrix. Matching coefficients in the certificate identity gives linear equality constraints on the Gram-matrix entries and the coefficients of the equality multipliers. Together with the positive semidefiniteness constraints, these define an SDP feasibility problem, which we solve numerically using MOSEK.

For exact certification, we extend the hybrid symbolic-numeric recovery procedure of Kaltofen et al. (2012), developed for unconstrained problems, to the constrained SOS identity above. Recovery proceeds sequentially: we first recover rational positive semidefinite Gram matrices for the inequality SOS multipliers $\sigma_1, \ldots, \sigma_m$ and keep these multipliers fixed. The remaining task is to recover σ_0 and the equality multipliers from $p - \sum_{i=1}^{m} g_i \sigma_i = \sigma_0 + \sum_{j=1}^{\ell} h_j q_j$. Gauss–Newton iterations refine the main SOS factorization and the equality multipliers to reduce the residual in this remaining identity. Rational recovery and exact coefficient correction then seek a rational main Gram matrix G_0 and rational equality multipliers satisfying the identity exactly. The complete polynomial identity and the positive semidefiniteness of all recovered Gram matrices are checked using exact arithmetic. Only certificates passing these checks are returned for subsequent Lean proof generation. Details are provided in Appendix B.1.

Bernstein-based proving under box constraints. For box-constrained goals, we use Bernstein representations and recursive subdivision, following Ray & Nataraj (2009). For a box

$$K = \prod_{i=1}^{n} [a_i, b_i] \quad \text{with} \quad a_i < b_i,$$

the affine substitution $x_i = a_i + (b_i - a_i) y_i$ maps the problem to the unit box $[0,1]^n$. We denote the transformed target polynomial by $\widetilde{p}$. Choose a multidegree bound $\mathbf{d} = (d_1, \ldots, d_n)$ such that $d_i \geq \deg_{y_i} \widetilde{p}$ for each i. The Bernstein basis polynomials are

$$B_{\mathbf{d},\alpha}(\mathbf{y}) = \prod_{i=1}^{n} \binom{d_i}{\alpha_i} y_i^{\alpha_i} (1 - y_i)^{d_i - \alpha_i}, \qquad 0 \leq \alpha_i \leq d_i. \tag{4}$$

We seek a representation $\widetilde{p} = \sum_\alpha c_\alpha B_{\mathbf{d},\alpha}$ with $c_\alpha \geq 0$, where the sum ranges over the indices in Eq. 4. Let A have the monomial coefficient vectors of these Bernstein polynomials as its columns, and let $\mathbf{v}$ be the coefficient vector of $\widetilde{p}$ in the same monomial basis. Matching coefficients yields the LP feasibility problem

$$\text{find } \mathbf{c} \quad \text{subject to} \quad A\mathbf{c} = \mathbf{v}, \qquad \mathbf{c} \geq 0. \tag{5}$$

Any feasible solution certifies $\widetilde{p} \geq 0$ on $[0,1]^n$, since every Bernstein basis polynomial is nonnegative on this domain. We first solve this feasibility problem in floating-point arithmetic to obtain a numerical candidate.

If the numerical solve does not yield a suitable candidate certificate, we select a variable and split its interval into two subintervals, leaving the intervals of all other variables unchanged, and recursively apply the procedure to both resulting subboxes. When a candidate is obtained, we treat its near-zero entries as tentative zeros. Fixing the corresponding coefficients to zero yields a reduced LP, which we solve in exact arithmetic. If exact re-solving succeeds, we check the complete coefficient-matching identity and the nonnegativity of all recovered coefficients against the original problem on the current box. A local certificate is accepted only after these checks pass. The original box is certified once every leaf box has an accepted certificate. The local certificates and the subdivision tree are recorded for subsequent Lean proof generation, including verification that the subboxes cover the original domain.

Successive difference substitution. For polynomial nonnegativity on the nonnegative orthant $K = \mathbb{R}^n_{\geq 0}$, DiffSub applies the successive difference substitution method described by Xia & Yang (2016, Chapter 10). For each permutation π of $\{1, \ldots, n\}$, consider the ordering $x_{\pi(1)} \geq \cdots \geq x_{\pi(n)} \geq 0$ and introduce nonnegative difference variables through

$$x_{\pi(i)} = \sum_{j=i}^{n} t_j, \qquad i = 1, \ldots, n, \qquad t_j \geq 0. \tag{6}$$

Here, $t_i = x_{\pi(i)} - x_{\pi(i+1)}$ for $i < n$, and $t_n = x_{\pi(n)}$. If the polynomial obtained by substituting into p has only nonnegative coefficients, its nonnegativity follows from $t_j \geq 0$, certifying the current ordering branch. Considering all variable orderings covers the entire domain K.

For every unresolved branch, the procedure repeats the ordering and substitution on the new nonnegative variables; certified branches require no further expansion. A complete certificate is obtained when every leaf branch is certified. All substitutions and coefficient checks use exact arithmetic, so no numerical recovery is required. The certificate records the ordering cases, substitution identities, and polynomial expansions at the leaves. If the search budget is exhausted while some branches remain unresolved, the backend returns an unsolved result.

The Translator converts each exact certificate into a local Lean proof. The agent integrates this proof into the surrounding context and justifies any reformulation connecting the polynomial statement to the original goal. A proof is accepted only after Lean verifies the original theorem statement.

4 Experiments

We evaluate IneqAgent on four inequality benchmarks, compare its proving performance with existing systems, and examine LLM inference cost across four backbones. We further present three case studies on conjectures concerning matrix permanents. These case studies are evaluated separately from the benchmark suite.

4.1 Experimental Setup

Benchmarks. The benchmark evaluation comprises 190 problems from three existing collections and our newly constructed LitIneq benchmark. MO-INT-20 is the 20-problem benchmark introduced with AIPS, comprising algebraic inequalities from the IMO, IMO shortlists, and national mathematical olympiads (Wei et al., 2024). Ineq-Real is the 50-problem subset of Ineq-Comp drawn from mathematics competitions, training materials, and Chinese Gaokao exams, with the problems manually formalized in Lean 4 (Zhao et al., 2025). The Evan Chen benchmark consists of 42 problems selected from *A Brief Introduction to Olympiad Inequalities*, Evan Chen's 2014 notes (Chen, 2014).

To extend the evaluation beyond competition-oriented inequalities, we construct LitIneq, a benchmark comprising 78 inequality-proving tasks based on examples from research papers on inequality proving and polynomial optimization, as well as selected inequality proof obligations from the Flyspeck formalization of the Kepler conjecture. For polynomial optimization examples, we formulate tasks that establish bounds on the objective functions under the original constraints. We formalize each task as a self-contained Lean 4 theorem statement. Appendix C documents the source, original theorem or example identifier, and formalization details for each problem. Results are reported separately for each benchmark. The conjecture case studies in Section 4.5 are evaluated separately and are excluded from the 190-problem aggregate.

Baselines and backbones. We compare with symbolic systems, formal provers, and proving agents. The symbolic baselines are Z3 (de Moura & Bjørner, 2008), Maple (Heck, 1993), and Isabelle/HOL's `sos` method (Chaieb & Meyer, 2025). The formal provers and agents are Goedel-Prover-V2-8B (Lin et al., 2025), DeepSeek-Prover-V2-7B (Ren et al., 2025), AxProverBase (Requena et al., 2026), LIPS (Li et al., 2025a), and Hilbert (Varambally et al., 2026).

IneqAgent uses DeepSeek-V4-Flash for the main benchmark comparison in Table 1. For the inference-cost comparison in Table 2, we evaluate IneqAgent and Direct Agent with four backbones: DeepSeek-V4-Flash, Qwen3.8-Flash, Kimi-K3, and GPT-5.5. Each pairwise comparison

uses the same backbone. Direct Agent denotes Codex running with its default agent configuration and access to Lean.

Evaluation protocol. All benchmark methods are evaluated with a two-hour time limit per problem. Success rates are computed over all problems in each benchmark, counting unsolved and timed-out problems as failures. Mean solving times are computed over the problems successfully solved by each method. For Z3 and Maple, success is assessed from solver outputs, and these runs do not produce proofs checked by a proof assistant. For IneqAgent, a problem is counted as solved only after Lean accepts the generated proof. Detailed evaluation settings and method-specific success criteria are provided in Appendix D.

4.2 Main Results on Inequality Benchmarks

Table 1 reports success rates and mean solving times. IneqAgent achieves the highest success rate on each of the four benchmarks, solving 19/20 problems in MO-INT-20, 46/50 in Ineq-Real, 42/42 in Evan Chen, and 71/78 in LitIneq. Overall, it solves 178/190 problems (93.68%), compared with 143/190 (75.26%) for Hilbert, the strongest baseline in overall success rate, yielding an improvement of 18.42 percentage points.

On LitIneq, IneqAgent solves 71 problems, compared with 57 for AxProverBase, the strongest baseline on this benchmark. This corresponds to a success-rate improvement of 17.95 percentage points, extending the observed advantage beyond competition-oriented inequalities to tasks drawn from research papers and Flyspeck.

Table 1: Success rate (SR, %) and mean solving time over problems successfully solved by each method (Time, seconds). Overall aggregates all 190 problems; its mean time is weighted by the number of successful problems in each benchmark. Bold indicates the highest success rate in each benchmark and overall. A dash indicates that no problem was solved.

	MO-INT-20		Ineq-Real		Evan Chen		LitIneq		Overall	
Method	SR	Time	SR	Time	SR	Time	SR	Time	SR	Time
Symbolic computation										
Z3	25.00	234.20	30.00	20.04	42.86	0.15	52.56	1.80	41.58	19.60
Maple	0.00	–	10.00	1.69	26.19	1.28	41.03	218.20	25.26	145.94
Isabelle `sos`	0.00	–	4.00	21.69	9.52	21.69	20.51	18.90	11.58	19.66
LLM provers										
Goedel-Prover-V2-8B	10.00	1323.86	28.00	779.01	30.95	842.92	58.97	834.75	39.47	838.80
DeepSeek-Prover-V2-7B	10.00	503.94	18.00	492.84	28.57	422.14	44.87	515.36	30.53	492.18
LLM-based proving frameworks										
AxProverBase	75.00	1908.00	70.00	1382.00	73.81	1315.00	73.08	463.00	72.63	1044.54
Hilbert	75.00	1947.28	86.00	2462.30	76.19	1849.04	67.95	842.54	75.26	1670.71
LIPS	35.00	2479.74	36.00	2496.18	54.76	2165.82	38.46	109.59	41.05	1479.37
IneqAgent (ours)	**95.00**	333.87	**92.00**	180.66	**100.00**	147.92	**91.03**	94.07	**93.68**	154.75

4.3 Proof Success under Cost Thresholds

We examine whether IneqAgent enables more problems to be proved at low LLM inference cost. Table 2 compares IneqAgent with Direct Agent on all 190 problems using four backbones. Each pairwise comparison uses the same backbone and pricing.

Cost accounting and thresholds. All costs are reported in US cents and measure LLM inference expenditure only, excluding the computational costs of symbolic solving and Lean verification. The cost of a run accumulates across its LLM calls, including unsuccessful attempts within that run. For each threshold $b \in \{5, 10, 50, 100\}$, we count the problems successfully solved with cumulative LLM inference cost at most b. Success rate uses all 190 problems as its denominator, while mean cost is computed separately over each method's qualifying successful runs. Unsolved and timed-out problems contribute to the success-rate denominator but are excluded from the mean cost. The No

cap setting includes all successful runs. These thresholds are applied post hoc rather than used as stopping budgets and every run remains subject to the two-hour time limit.

Table 2: Proof success and LLM inference cost on 190 problems. Both methods use the backbone specified in the first column. All thresholds and mean costs are in US cents. SR (%) is the percentage of all 190 problems solved with cumulative LLM inference cost at or below the specified threshold; Cost is the mean cost over those successful runs. Thresholds are applied post hoc. No cap includes all successful runs and the two-hour per-problem limit still applies. Within each backbone and threshold, the highest SR is bold; the lower Cost is bold when both means are defined.

		LLM inference cost threshold (US cents)									
		5		10		50		100		No cap	
Backbone	Method	SR	Cost	SR	Cost	SR	Cost	SR	Cost	SR	Cost
DeepSeek-V4-Flash	Direct Agent	65.79	1.9503	78.42	2.8013	89.47	4.6829	89.47	4.6829	89.47	4.6829
	IneqAgent (ours)	**87.89**	**0.6597**	**92.11**	**0.9272**	**93.68**	**1.3340**	**93.68**	**1.3340**	**93.68**	**1.3340**
Qwen3.8-Flash	Direct Agent	67.89	1.7879	83.68	2.7165	91.05	3.8919	**92.11**	4.5606	**92.11**	4.5606
	IneqAgent (ours)	**83.68**	**0.7979**	**88.95**	**1.1459**	**91.58**	**1.4699**	91.58	**1.4699**	91.58	**1.4699**
Kimi-K3	Direct Agent	6.84	3.6009	25.26	6.4002	63.16	17.3555	76.32	26.0644	**91.58**	57.2108
	IneqAgent (ours)	**51.58**	**0.1257**	**55.79**	**0.6654**	**76.32**	**7.8493**	**81.58**	**11.2559**	85.79	**18.4748**
GPT-5.5	Direct Agent	0.00	–	2.11	8.70	57.89	31.38	**77.37**	41.37	**90.00**	71.44
	IneqAgent (ours)	**50.00**	0.1193	**51.58**	**0.3754**	**67.89**	**6.6971**	75.26	**12.8798**	**90.00**	**61.2678**

IneqAgent achieves higher success rates at the 5-, 10-, and 50-cent thresholds across four backbones. At 5 cents, the improvements range from 15.79 to 50.00 percentage points. For instance, DeepSeek-V4-Flash improves from 65.79% to 87.89%, while GPT-5.5 improves from 0.00% to 50.00%. These results show that more problems can be solved at low LLM inference cost with IneqAgent.

The relative success rates vary at higher thresholds. At 100 cents, Direct Agent achieves higher success rates with Qwen3.8-Flash and GPT-5.5. Without a cost cap, IneqAgent achieves a higher success rate with DeepSeek-V4-Flash, matches Direct Agent with GPT-5.5, and has lower success rates with Qwen3.8-Flash and Kimi-K3. Mean inference costs over each method's successful runs are lower for IneqAgent across all four backbones. For Qwen3.8-Flash, the mean decreases from 4.5606 to 1.4699 cents, and for GPT-5.5, it decreases from 71.44 to 61.2678 cents. These means describe each method's own successful runs and do not constitute comparisons on an identical set of problems.

4.4 Ablation Study

We ablate lemma-level MCTS and symbolic computation using DeepSeek-V4-Flash on the same 190 problems, with a two-hour time limit per problem. The *Base agent* generates and revises proofs with Lean feedback, using neither component. *Base + MCTS* adds lemma-level tree search, while *Base + Symbolic* enables symbolic proving of the original goal and intermediate lemmas. *IneqAgent (full)* combines both components.

Table 3: Ablation results with DeepSeek-V4-Flash on all 190 benchmark problems under a two-hour time limit per problem. Mean LLM inference cost and wall-clock time are computed over each variant's successful problems. Bold indicates the highest success rate or the lowest reported mean.

Variant	SR (%) ↑	Avg. LLM cost (US cents) ↓	Avg. time (s) ↓
Base agent	89.47	4.6829	336.24
Base + MCTS	91.05	2.2765	196.97
Base + Symbolic	90.53	2.7809	198.16
IneqAgent (full)	**93.68**	**1.3340**	**154.75**

Table 3 shows that both components improve performance when added individually. MCTS increases the number of solved problems from 170 to 173, while symbolic computation increases it to 172. Combining both components solves 178 problems, five more than MCTS alone and six more

than symbolic computation alone. These results support the benefits of combining tree search with symbolic computation in IneqAgent.

The full configuration also achieves the lowest mean LLM inference cost and mean wall-clock time among the four variants. Relative to the base agent, these means are lower by 71.51% and 53.98%, respectively. Both measures are averaged over each variant's own successful problems.

4.5 Applications to Conjectures on Matrix Permanents

We evaluate IneqAgent on three polynomial proof tasks arising from conjectures on matrix permanents: strengthened forms of Dittert's conjecture (Minc, 1983) in dimensions four and five, and the Holens–Djokovic $(4,4)$ inequality (Holens, 1964; Djoković, 1967), which implies the four-dimensional case of Merris's conjecture (Merris, 1973). These case studies examine the framework's ability to combine agentic reasoning with exact symbolic certification on research-level mathematical problems. They are evaluated separately from the 190-problem benchmark suite and are not subject to its two-hour time limit. Task sizes and costs are reported in Table 9 in Appendix E.

Dittert's conjecture. For a nonnegative $n \times n$ matrix $A = (a_{ij})$ with $\sum_{i,j} a_{ij} = n$, let $r_i = \sum_j a_{ij}$ and $c_j = \sum_i a_{ij}$ denote its row and column sums. Writing $\mathrm{per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$, define $\Phi_n(A) = \prod_i r_i + \prod_j c_j - \mathrm{per}(A)$. Dittert's conjecture asserts that $\Phi_n(A) \le 2 - n!/n^n$, with equality if and only if $A = J_n$, where every entry of J_n equals $1/n$. For $n \in \{4,5\}$, IneqAgent produces Lean-verified proofs of the strengthened polynomial inequalities

$$p_n(A) := 2 - \frac{n!}{n^n} - \Phi_n(A) - \kappa_n \|A - J_n\|_F^2 \ge 0, \tag{7}$$

under the same constraints, where $\kappa_4 = 1/52$, $\kappa_5 = 1/2500$, and $\|\cdot\|_F$ denotes the Frobenius norm. These bounds establish quantitative stability and the unique maximizer J_n in both dimensions. For certificate search, we use the scaled target $P_4 = 32 p_4$ in dimension four and a homogeneous reformulation $\widetilde{F}_5$ in dimension five, defined in Appendix E.

Holens–Djokovic $(4,4)$ and Merris's conjecture. Let Ω_4 denote the set of 4×4 nonnegative matrices whose row and column sums equal one. The Holens–Djokovic $(4,4)$ inequality is the polynomial nonnegativity statement

$$Q_4(A) := 24\,\mathrm{per}(A) - 2 - \sum_{i,j=1}^{4} a_{ij}^3 \ge 0, \qquad A \in \Omega_4. \tag{8}$$

Expressing the remaining entries in terms of the upper-left 3×3 block eliminates the row- and column-sum equalities, yielding a degree-four polynomial problem in nine variables while retaining the nonnegativity constraints on all entries. IneqAgent constructs an exact rational SOS certificate for the reduced problem.

This result implies Merris's conjecture in dimension four. Let $S_j = \sum_{i=1}^{4} \mathrm{per}(A(i \mid j))$, where $A(i \mid j)$ is obtained by deleting row i and column j. For $A \in \Omega_4$,

$$\min_{1 \le j \le 4} S_j \le \frac{1}{4} \sum_{j=1}^{4} S_j = 4\,\mathrm{per}(A) - \frac{1}{6} Q_4(A) \le 4\,\mathrm{per}(A).$$

The SOS certificate and the case in which S_4 is minimal are verified in Lean; column relabeling gives the conclusion for any minimizing column.

5 Conclusion

We introduced IneqAgent, an agentic framework for formal inequality proving in Lean 4. Through lemma-level MCTS, it coordinates LLM reasoning, goal reformulation, and symbolic computation, integrating certificate-derived local proofs into complete proofs checked against the original theorem statements. Evaluation on four benchmarks and three polynomial proof tasks arising from matrix-permanent conjectures demonstrates its proving capability. Across four backbones, IneqAgent achieves higher success rates at low LLM inference cost thresholds than Direct Agent using the same backbone, with lower mean inference costs over each method's successful runs.

AI USE STATEMENT

We used generative AI tools to polish the writing, assist in finding relevant literature and reference materials, and carry out experiments according to author-specified instructions, including which tests to run and how to run them. We reviewed the AI-assisted text, consulted the cited sources, and checked the experimental outputs. We take full responsibility for the content and results of this work.

REFERENCES

Amine Chaieb and Philipp Meyer. Sum of Squares: A decision procedure for universal multivariate real arithmetic using semidefinite programming. Isabelle/HOL Library, theory Sum_of_Squares, 2025. URL https://isabelle.in.tum.de/website-Isabelle2025-2/dist/library/HOL/HOL-Library/Sum_of_Squares.html.

Evan Chen. A brief introduction to olympiad inequalities. https://web.evanchen.cc/handouts/Ineq/en.pdf, 2014.

Gi-Sang Cheon and Ian M. Wanless. An update on Minc's survey of open problems involving permanents. *Linear Algebra and its Applications*, 403:314–342, 2005. doi: 10.1016/j.laa.2005.02.030.

Gi-Sang Cheon and Ian M. Wanless. Some results towards the Dittert conjecture on permanents. *Linear Algebra and its Applications*, 436(4):791–801, 2012. doi: 10.1016/j.laa.2010.08.041.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction—CADE 28*, pp. 625–635. Springer, 2021.

D. Ž. Djoković. On a conjecture by van der Waerden. *Matematički Vesnik*, 4(19)(4):272–276, 1967.

André Heck. *Introduction to MAPLE*. Springer, 1993. doi: 10.1007/978-1-4684-0519-4.

F. Holens. Two aspects of doubly stochastic matrices: Permutation matrices and the minimum of the permanent function. Master's thesis, University of Manitoba, 1964.

Suk Geun Hwang. A note on a conjecture on permanents. *Linear Algebra and its Applications*, 76:31–44, 1986. doi: 10.1016/0024-3795(86)90212-0.

Suk Geun Hwang. On a conjecture of E. Dittert. *Linear Algebra and its Applications*, 95:161–169, 1987. doi: 10.1016/0024-3795(87)90032-2.

Erich L. Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. *Journal of Symbolic Computation*, 47(1):1–15, 2012.

Guillaume Lample, Timothée Lacroix, Marie-Anne Lachaux, Aurélien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in Neural Information Processing Systems*, 35:26337–26349, 2022.

Zenan Li, Zhaoyu Li, Wen Tang, Xian Zhang, Yuan Yao, Xujie Si, Fan Yang, Kaiyu Yang, and Xiaoxing Ma. Proving olympiad inequalities by synergizing LLMs and symbolic reasoning. In *International Conference on Learning Representations*, pp. 2304–2329, 2025a.

Zhaoqun Li, Beishui Liao, and Qiwei Ye. IneqSearch: Hybrid reasoning for olympiad inequality proofs. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6c92839f0f9cddc96c694712a7143b09-Abstract-Conference.html.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025.

Banglong Liu, Niuniu Qi, Xia Zeng, Lydia Dehbi, and Zhengfeng Yang. Automated proof of polynomial inequalities via reinforcement learning. *arXiv preprint arXiv:2503.06592*, 2025. URL https://arxiv.org/abs/2503.06592.

Victor Magron and Mohab Safey El Din. RealCertify: A Maple package for certifying nonnegativity. *ACM Communications in Computer Algebra*, 52(2):34–37, 2018.

Marvin Marcus and Russell Merris. A relation between the permanental and determinantal adjoints. *Journal of the Australian Mathematical Society*, 15(3):270–271, 1973. doi: 10.1017/S1446788700013173.

Russell Merris. The permanent of a doubly stochastic matrix. *The American Mathematical Monthly*, 80(7):791–793, 1973. doi: 10.1080/00029890.1973.11993372.

Henryk Minc. Theory of permanents 1978–1981. *Linear and Multilinear Algebra*, 12:227–263, 1983.

César Muñoz and Anthony Narkawicz. Formalization of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, 2013. doi: 10.1007/s10817-012-9256-3. URL https://doi.org/10.1007/s10817-012-9256-3.

Zhekai Pang. Proof of Dittert's conjecture for dimensions $n \geq 17$. *arXiv preprint arXiv:2606.01531*, 2026. URL https://arxiv.org/abs/2606.01531.

Helfried Peyrl and Pablo A. Parrilo. Computing sum of squares decompositions with rational coefficients. *Theoretical Computer Science*, 409(2):269–281, 2008.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

Shashwati Ray and P. S. V. Nataraj. An efficient algorithm for range computation of polynomials using the Bernstein form. *Journal of Global Optimization*, 45(3):403–426, 2009. doi: 10.1007/s10898-008-9382-y. URL https://doi.org/10.1007/s10898-008-9382-y.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Borja Requena, Austin Letson, Krystian Nowakowski, Izan Beltran-Ferreiro, and Leopoldo Sarra. A minimal agent for automated theorem proving. *arXiv preprint arXiv:2602.24273*, 2026.

Richard Sinkhorn. A problem related to the van der Waerden permanent theorem. *Linear and Multilinear Algebra*, 16(1–4):167–173, 1984. doi: 10.1080/03081088408817620.

Divya K. Udayan and Kanagasabapathi Somasundaram. The inequalities of Merris and Foregger for permanents. *Symmetry*, 13(10):1782, 2021. doi: 10.3390/sym13101782.

Divya K. Udayan and Kanagasabapathi Somasundaram. Lih Wang's and Dittert's conjectures on permanents. *Special Matrices*, 12(1):20240006, 2024. doi: 10.1515/spma-2024-0006.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.11354.

Chenrui Wei, Mengzhou Sun, and Wei Wang. Proving olympiad algebraic inequalities without human demonstrations. *Advances in Neural Information Processing Systems*, 37, 2024.

Bican Xia and Lu Yang. *Automated Inequality Proving and Discovering*. World Scientific, 2016.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, 2023.

Haoyu Zhao, Yihan Geng, Shange Tang, Yong Lin, Bohan Lyu, Hongzhou Lin, Chi Jin, and Sanjeev Arora. Ineq-Comp: Benchmarking human-intuitive compositional reasoning in automated theorem proving on inequalities. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.12680. Datasets and Benchmarks Track.

Ruobing Zuo, Hanrui Zhao, Gaolei He, Zhengfeng Yang, and Jianlin Wang. From LLM-generated conjectures to Lean formalizations: Automated polynomial inequality proving via sum-of-squares certificates. OpenReview: https://openreview.net/forum?id=Lz8rHDmmKt, 2026. URL https://arxiv.org/abs/2605.15445. Accepted to ICML 2026.

A LEMMA-LEVEL SEARCH DETAILS

This appendix gives the search rules used to allocate proving attempts in Section 3.1. Each node v represents a partial proof state s_v, comprising the local context, established facts, and remaining goals. An accepted edge $v \to v'$ records a proof extension with one or more fully proved local lemmas. Paths represent successive extensions, while sibling nodes retain alternative ways to advance the same state.

Selection and progressive widening. MCTS allocates proving attempts across alternative proof paths. For each node v, we maintain a visit count $N(v)$ and accumulated reward $W(v)$ from attempts backed up through that node. Their ratio $Q(v) = W(v)/N(v)$ is the mean reward, initialized to zero for unvisited nodes. A stagnation count $S(v)$ tracks consecutive attempts to expand v without useful progress. An open node is a valid proof state with remaining goals. At each iteration, selection starts at the root and either expands the current node or descends to an open child. Progressive widening sets the threshold for adding children:

$$B(v) = \max\{1, \lfloor k \max(1, N(v))^{\alpha} \rfloor + \delta(S(v))\}, \tag{9}$$

where k sets the branching scale, $\alpha \in (0, 1)$ controls its growth, and $\delta(S) = \min\{2, \lfloor S/2 \rfloor\}$ allows up to two additional children when progress stalls. Only accepted proof states count toward this child limit; failed attempts are recorded separately. If v has fewer than $B(v)$ children, the agent generates a new extension from v. Otherwise, it descends to the open child u maximizing the selection score

$$U(u \mid v) = Q(u) + c\sqrt{\frac{\log(N(v)+1)}{N(u)+1}} - \min\{\rho S(u), \rho_{\max}\}. \tag{10}$$

Here $Q(u)$ favors branches with higher observed rewards, the exploration term weighted by c encourages less-visited branches, and the bounded penalty discourages repeatedly unproductive ones, with strength ρ and maximum value $\rho_{\max}$. Selection continues until a node can be expanded. If no open child is available, the agent attempts a new extension at the current node, even if the widening threshold has been reached.

Search parameters. Table 4 gives the default search configuration. We use ρ for the stagnation penalty and reserve λ for the Lean–LLM reward weight. New nodes initialize $N = W = Q = S = 0$. Selection scores are rounded to three decimal places.

Table 4: Default parameters for lemma-level search.

Parameter	Meaning	Value
k	Progressive-widening scale	1.0
α	Progressive-widening exponent	0.5
c	Exploration coefficient	1.414
λ	Weight of the Lean-based reward	0.5
ρ	Stagnation penalty per attempt	0.06
$\rho_{\max}$	Maximum stagnation penalty	0.30

Reward and backpropagation. After expansion, Lean verification determines the candidate's status: `solved` if it completes the theorem without `sorry` or `admit`, `open` if its intermediate lemmas are verified but goals remain, or `failed` if verification fails. Failed attempts provide diagnostics for repair; only valid open states can be selected for further search. For an open candidate v', we combine a Lean-based heuristic score $r_{v'}^{\text{Lean}} \in [0, 1]$ with an LLM score $r_{v'}^{\text{LLM}} \in [0, 1]$. The former uses Lean checking results and proof-text features defined below. The latter compares the parent and candidate goals to estimate whether the new state is simpler and closer to completing the theorem. The LLM score guides search; correctness is determined by Lean. The reward is

$$r_{v'} = \begin{cases} 1, & \text{if } v' \text{ is solved,} \\ 0, & \text{if } v' \text{ fails Lean verification,} \\ \lambda r_{v'}^{\text{Lean}} + (1-\lambda) r_{v'}^{\text{LLM}}, & \text{if } v' \text{ is open,} \end{cases} \tag{11}$$

where $\lambda = 0.5$ gives equal weight to the two progress signals for an open candidate. The terminal rules take precedence: solved candidates receive one and failed candidates receive zero, regardless of the heuristic components. This final reward, rather than either component alone, updates the search statistics.

A valid extension creates a child, and its reward is backed up from that child to the root. A failed extension creates no accepted child; its diagnostics are retained, and its zero reward is backed up from the expanded parent v to the root. Thus, failed attempts increase visit counts without adding reward or occupying a child slot. Each node u on the corresponding path is updated once:

$$N(u) \leftarrow N(u) + 1, \qquad W(u) \leftarrow W(u) + r_{v'}, \qquad Q(u) \leftarrow W(u)/N(u). \tag{12}$$

Thus, $Q(u)$ summarizes rewards from attempts through the branch rooted at u, rather than assigning a fixed score to its current goals. Search returns a complete Lean-verified proof when one is found and otherwise terminates when the budget is exhausted or no open state remains.

Lean-based heuristic score. Each candidate undergoes a compilation check allowing unfinished `sorry`/`admit` markers and a completion check requiring a Lean-accepted proof without these markers. For the heuristic, let e indicate whether the proof text differs from its parent's, let $m_v, m_{v'}$ count occurrences of the unfinished markers in the two texts, and let $\Delta = m_v - m_{v'}$. Let b indicate successful compilation. These marker counts are textual features, rather than counts of Lean's remaining goals.

Let a count newly added lines beginning with `have`, `suffices`, a typed `let`, `lemma`, or `theorem`. Added lines are candidate lines absent from the parent text. These features measure proof structure; local proof validity is determined separately by Lean. Starting from $h = 0$, we add the applicable terms in Table 5 and set

$$r_{v'}^{\text{Lean}} = \text{round}_3(\min\{0.95, \max\{0, h\}\}) \,. \tag{13}$$

The structured-step bonus and the compilable-edit bonus are mutually exclusive. The agent timeout and execution-error penalties are also mutually exclusive; other applicable terms accumulate. The heuristic table includes failed-candidate features, but the failed branch of equation 11 overrides their sum with zero.

Table 5: Additive terms in the Lean-based heuristic. Agent execution refers to the proving-agent invocation for the current expansion.

Condition	Contribution to h
Candidate compiles ($b = 1$)	$+0.35$
Proof text changed ($e = 1$)	$+0.10$
Proof text unchanged ($e = 0$)	-0.12
Change in unfinished-marker count	$\max\{-0.30, \min\{0.30, 0.18\Delta\}\}$
$m_{v'} > 0$ unfinished markers remain	$-\min\{0.20, 0.04 m_{v'}\}$
$a > 0$ new structured steps	$+\min\{0.20, 0.05a\}$
$a = 0$, $e = 1$, and $b = 1$	$+0.03$
Candidate status is `failed`	-0.25
Agent times out, with / without a text change	-0.20 / -0.35
Agent execution errors without a timeout	-0.10
Either Lean check times out (charged once)	-0.20
Proof text changed but does not compile	$+0.05$

LLM progress score. For an open candidate, the evaluator receives the target theorem, the parent and candidate proofs, and the candidate's Lean checking result. It assesses the change in the proof rather than the absolute quality of the candidate. The prompt is given below; the placeholders are filled with the corresponding theorem, proof texts, and checking output. The numeric `score` is parsed and checked to lie in $[0, 1]$ before combining it with r^{Lean}. The heuristic's numeric value is not included in the prompt.

Prompt Template for LLM Progress Evaluation

Task:
You are evaluating one step in a Lean proof search. Compare the child proof with the parent proof for the target theorem. Score the substantive progress made in this step, not the child's absolute quality.

Instructions:
Give credit for resolving proof obligations or adding useful intermediate results. Do not reward longer text, cosmetic edits, unsupported claims, or removing `sorry` without a valid proof. Use the Lean result to check whether claimed progress is credible.

Scoring:
Score from 0 to 1:
0 no progress or regression;
0.25 small credible step;
0.5 substantial step;
0.75 most remaining work resolved;
1 theorem completely proved.
When evidence is unclear, score conservatively.

Input:
Target theorem: {`theorem_statement`}
Parent proof: {`parent_content`}
Child proof: {`child_content`}
Child Lean result: {`lean_result`}

Output format:
Return only JSON:
`{"score": 0.0, "reason": "one concise sentence"}`

Stagnation updates. Only a direct attempt to expand v updates $S(v)$; reward backpropagation does not change the stagnation counts of its ancestors. Let $M(v, v')$ be true if the candidate solves the theorem, or if its text changes and at least one of the following holds: $\Delta > 0$; it compiles and adds a structured step; or it compiles and has status `open`. Thus, the marker-reduction condition does not require compilation, while a changed, compilable open candidate satisfies this progress test even without reducing markers or adding structured steps. Using the final reward $r_{v'}$ and the parent's mean $Q_{\text{old}}(v)$ before backpropagation, we update

$$S(v) \leftarrow \begin{cases} 0, & \text{if } M(v, v') \text{ or } r_{v'} > Q_{\text{old}}(v) + 0.03, \\ S(v) + 1, & \text{otherwise.} \end{cases} \tag{14}$$

This local count determines both the widening slack and the bounded selection penalty. The next search iteration uses the updated counts and backed-up rewards.

B Details of Symbolic Proving

This appendix details the three symbolic proving backends introduced in Section 3.2. We consider polynomial nonnegativity goals of the form in Eq. 2, with $p, g_i, h_j \in \mathbb{Q}[\mathbf{x}]$ and variables ranging over $\mathbb{R}^n$. For box-constrained goals, the interval endpoints are also rational. Each backend searches for an exact certificate for a polynomial goal supplied by the agent, either directly from the input theorem or through an intermediate lemma or reformulation. Successful certificates are translated into local Lean proofs, and unsuccessful attempts return feedback for further agentic reasoning.

B.1 SOS-Based Proving under Polynomial Constraints

Degree-bounded SDP formulation. We use the polynomial goal and domain K from Section 3.2. The backend seeks a constrained SOS identity

$$p = \sigma_0 + \sum_{i=1}^{m} g_i \sigma_i + \sum_{j=1}^{\ell} h_j q_j, \tag{15}$$

where each σ_i is a sum of squares and each q_j is a polynomial multiplier unrestricted in sign. This certificate format is motivated by Putinar's Positivstellensatz (Putinar, 1993). The theorem's existence guarantee requires strict positivity and an Archimedean quadratic module, modulo the equality ideal when equalities are present. Our use of the certificate relies on its sufficiency: any exact identity of this form proves $p \geq 0$ on K.

Set $g_0 = 1$ and omit identically zero constraint polynomials. Choose a relaxation order d such that $\deg p, \deg g_i, \deg h_j \leq 2d$, and define

$$d_i = \left\lfloor \frac{2d - \deg g_i}{2} \right\rfloor, \qquad e_j = 2d - \deg h_j. \tag{16}$$

Let z_i contain all monomials of total degree at most d_i in a fixed order. We represent $\sigma_i = z_i^{\mathsf{T}} G_i z_i$ with $G_i \succeq 0$ and impose $\deg q_j \leq e_j$. The resulting identity is

$$p = \sum_{i=0}^{m} g_i z_i^{\mathsf{T}} G_i z_i + \sum_{j=1}^{\ell} h_j q_j, \qquad G_i \succeq 0. \tag{17}$$

Every summand has degree at most $2d$. Let coeff_{2d} extract coefficients in the monomial basis of total degree at most $2d$, and let $\mathcal{A}_d(G, q)$ be the coefficient vector of the right-hand side. Coefficient matching gives the SDP feasibility problem

$$\begin{aligned} &\text{find } G_0, \ldots, G_m, q_1, \ldots, q_\ell \quad \text{such that} \\ &\mathcal{A}_d(G, q) = \mathrm{coeff}_{2d}(p), \qquad G_i \succeq 0 \quad (i = 0, \ldots, m). \end{aligned} \tag{18}$$

Here the decision variables are the independent Gram-matrix entries and the coefficients of the q_j. MOSEK supplies numerical candidates G_b^{num}, where $b = 0$ indexes the main SOS block and $b = 1, \ldots, m$ indexes the SOS multipliers of the inequality constraints.

Rational recovery of SOS multipliers. We extend the numerical–symbolic certification procedure of Kaltofen et al. (2012), developed for unconstrained problems, to the constrained SOS identity in Eq. 15. We recover the certificate sequentially, first fixing the SOS multipliers $\sigma_1, \ldots, \sigma_m$ and then recovering the main SOS block together with the equality multipliers. For a candidate denominator D, numerical matrix entries are rounded to multiples of $1/D$ and symmetrized in exact arithmetic. Each rational candidate is checked by an exact LDL^{T} decomposition for positive semidefiniteness. The denominator is adjusted if a candidate fails this check.

For singular blocks with an available exact rational kernel subspace, recovery can be performed in reduced coordinates. Let the columns of P_b form a rational basis for the orthogonal complement of that subspace, and define

$$L_b = (P_b^{\mathsf{T}} P_b)^{-1} P_b^{\mathsf{T}}, \qquad S_b^{\mathrm{num}} = L_b G_b^{\mathrm{num}} L_b^{\mathsf{T}}. \tag{19}$$

Rational recovery and the exact positivity check are then applied to S_b^{num}. An accepted $S_b^{\mathbb{Q}} \succeq 0$ gives $G_b^{\mathbb{Q}} = P_b S_b^{\mathbb{Q}} P_b^{\mathsf{T}} \succeq 0$, preserving the prescribed kernel subspace. This reduction is used only when the required exact kernel information is available.

After recovering the inequality-multiplier blocks, set

$$\sigma_i = z_i^{\mathsf{T}} G_i^{\mathbb{Q}} z_i \quad (i = 1, \ldots, m), \qquad C_{\mathrm{sm}} := \sum_{i=1}^{m} g_i \sigma_i, \qquad R_0 := p - C_{\mathrm{sm}}. \tag{20}$$

These rational SOS multipliers remain fixed throughout the subsequent recovery. The remaining task is

$$R_0 = \sigma_0 + \sum_{j=1}^{\ell} h_j q_j. \tag{21}$$

Thus only one SOS block remains to be recovered; the equality multipliers are still to be determined. With no equality constraints, the task reduces to an SOS decomposition of R_0.

Main-block recovery and exact verification. For the remaining main block, we use Gauss–Newton refinement followed by rational recovery. Write $\bar{z}_0 = z_0$ without reduction, or $\bar{z}_0 = P_0^{\mathsf{T}} z_0$ when using reduced coordinates. Factor G_0^{num}, or S_0^{num} in reduced coordinates, approximately as $U^{\mathsf{T}}U$. If θ collects the entries of U and the coefficients of the equality multipliers, define

$$\mathcal{R}(\theta) = \mathrm{coeff}_{2d}\left(R_0 - \|U\bar{z}_0\|_2^2 - \sum_{j=1}^{\ell} h_j q_j\right). \tag{22}$$

At iteration k, the Gauss–Newton update solves the linearized least-squares problem

$$\begin{aligned} \Delta\theta_k &\in \operatorname*{argmin}_{\Delta\theta} \left\|\mathcal{R}(\theta_k) + J_{\mathcal{R}}(\theta_k)\Delta\theta\right\|_2^2, \\ \theta_{k+1} &= \theta_k + \Delta\theta_k, \end{aligned} \tag{23}$$

where $J_{\mathcal{R}}$ is the residual Jacobian. The previously recovered SOS multipliers enter only through the fixed polynomial R_0.

From the refined matrix $U^{\mathsf{T}}U$, rational reconstruction and exact coefficient correction seek a main Gram matrix $G_0^{\mathbb{Q}}$ and rational equality multipliers satisfying

$$R_0 = z_0^{\mathsf{T}} G_0^{\mathbb{Q}} z_0 + \sum_{j=1}^{\ell} h_j q_j, \qquad G_0^{\mathbb{Q}} \succeq 0, \qquad q_j \in \mathbb{Q}[\mathbf{x}]. \tag{24}$$

In reduced coordinates, the recovered matrix is $S_0^{\mathbb{Q}}$ and $G_0^{\mathbb{Q}} = P_0 S_0^{\mathbb{Q}} P_0^{\mathsf{T}}$. Recovery targets the Gram matrix; the numerical factor U need not itself have rational entries. The coefficient correction enforces the identity in Eq. 24, and positive semidefiniteness is checked again after correction. For a single affine equality $h = 0$, the equality multiplier can instead be obtained by exact division of $R_0 - \sigma_0$ by h, with zero remainder required. If recovery or verification fails, recovery parameters are adjusted and the main block is refined or recomputed, while the accepted SOS multipliers remain fixed. Exhausting the available budget returns an unsolved result.

Finally, the complete identity in Eq. 15 is verified coefficientwise over $\mathbb{Q}$. Exact LDL^{T} decompositions supply weighted-square representations

$$\sigma_b = \sum_k w_{bk} s_{bk}(\mathbf{x})^2, \qquad w_{bk} \in \mathbb{Q}_{\geq 0}, \quad s_{bk} \in \mathbb{Q}[\mathbf{x}]. \tag{25}$$

Together with the rational equality multipliers, these representations form the certificate returned for Lean proof generation. On K, all SOS terms and their products with g_i are nonnegative, while every $h_j q_j$ vanishes, establishing $p \geq 0$.

B.2 Bernstein-Based Proving under Box Constraints

This backend uses Bernstein representations and recursive subdivision to certify polynomial nonnegativity on a box, following Ray & Nataraj (2009). The numerical candidate search and exact recovery procedure are described below.

Coefficient matching on a box. Let $K = \prod_{i=1}^{n}[a_i, b_i]$, with rational endpoints $a_i < b_i$. The affine map $T_K(\mathbf{y})_i = a_i + (b_i - a_i)y_i$ maps $[0,1]^n$ onto K. Write $\widetilde{p}_K = p \circ T_K$, and choose a multidegree $\mathbf{d} = (d_1, \ldots, d_n)$ with $d_i \geq \deg_{y_i} \widetilde{p}_K$. Define $I_{\mathbf{d}} = \{\alpha \in \mathbb{Z}_{\geq 0}^n : \alpha_i \leq d_i\}$. The Bernstein basis polynomials in Eq. 4 are nonnegative on $[0,1]^n$. Thus an expansion

$$\widetilde{p}_K(\mathbf{y}) = \sum_{\alpha \in I_{\mathbf{d}}} c_\alpha B_{\mathbf{d},\alpha}(\mathbf{y}), \qquad c_\alpha \geq 0, \tag{26}$$

certifies $p \geq 0$ on K.

Write $\widetilde{p}_K = \sum_{\beta \in I_{\mathbf{d}}} v_\beta \mathbf{y}^\beta$, including zero coefficients where necessary. Let A contain the monomial coefficient vectors of the Bernstein polynomials as its columns. Its entries are

$$A_{\beta,\alpha} = \begin{cases} \prod_{i=1}^{n} \binom{d_i}{\alpha_i} \binom{d_i - \alpha_i}{\beta_i - \alpha_i} (-1)^{\beta_i - \alpha_i}, & \alpha \le \beta, \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

where multi-index inequalities are componentwise and both indices range over $I_{\mathbf{d}}$. Matching coefficients gives the LP feasibility problem

$$\text{find } \mathbf{c} \quad \text{subject to} \quad A\mathbf{c} = \mathbf{v}, \qquad \mathbf{c} \ge 0. \tag{28}$$

For a fixed multidegree, the Bernstein coefficients are unique, so this LP tests whether the unique coefficient vector is nonnegative.

Numerical candidates and exact recovery. We first solve Eq. 28 in floating-point arithmetic. Given a numerical candidate $\widehat{\mathbf{c}}$ and a zero threshold $\tau > 0$, we form the tentative support $S = \{\alpha \in I_{\mathbf{d}} : |\widehat{c}_\alpha| > \tau\}$. Fixing the remaining coefficients to zero gives the reduced problem

$$\text{find } \mathbf{u} \in \mathbb{Q}^{|S|} \quad \text{subject to} \quad A_{:,S}\mathbf{u} = \mathbf{v}, \qquad \mathbf{u} \ge 0, \tag{29}$$

which we solve in exact arithmetic. Here $A_{:,S}$ contains the columns indexed by S. We reconstruct the full vector $\mathbf{c}^{\mathbb{Q}}$ by assigning the recovered coefficients to S and zero to its complement, then verify

$$A\mathbf{c}^{\mathbb{Q}} = \mathbf{v}, \qquad \mathbf{c}^{\mathbb{Q}} \ge 0 \tag{30}$$

exactly. Only a candidate passing both checks certifies the current box. The numerical threshold proposes a support; it does not establish that the discarded coefficients are exactly zero. If exact recovery or verification fails, the attempt returns an unresolved result to the search controller.

Binary subdivision and certificate construction. If no suitable numerical candidate is obtained and the subdivision budget permits, we choose a coordinate k and a rational split point $s \in (a_k, b_k)$, and form

$$K^- = K \cap \{\mathbf{x} : x_k \le s\}, \qquad K^+ = K \cap \{\mathbf{x} : x_k \ge s\}. \tag{31}$$

This replaces $[a_k, b_k]$ by $[a_k, s]$ and $[s, b_k]$, leaving all other coordinate intervals unchanged. Each child box is normalized separately, and the same certificate search is applied recursively to the original polynomial p on that box. Since $K = K^- \cup K^+$, certificates for both children certify the parent; the shared boundary is included in both closed boxes.

A successful certificate consists of the binary subdivision tree and an exact nonnegative Bernstein expansion for every leaf box. The Translator uses these data to verify the affine substitutions, local polynomial identities, coefficient nonnegativity, and coverage of the original box in Lean. If the search budget is exhausted while any leaf remains unresolved, the backend returns an unsolved result.

B.3 Successive Difference Substitution

DiffSub searches for a finite certificate of polynomial nonnegativity on $\mathbb{R}^n_{\ge 0}$ using successive difference substitution (Xia & Yang, 2016, Chapter 10). The procedure combines variable ordering with linear substitutions and certifies branches whose transformed polynomials have nonnegative coefficients.

Ordering domains and substitutions. Let $\mathfrak{S}_n$ denote the permutations of $\{1, \dots, n\}$. For each $\pi \in \mathfrak{S}_n$, define

$$C_\pi = \{\mathbf{x} \in \mathbb{R}^n_{\ge 0} \mid x_{\pi(1)} \ge \cdots \ge x_{\pi(n)}\}. \tag{32}$$

Every nonnegative tuple admits such an ordering, so $\bigcup_{\pi \in \mathfrak{S}_n} C_\pi = \mathbb{R}^n_{\ge 0}$. On each ordering domain, introduce nonnegative difference variables through the linear map

$$(T_\pi \mathbf{t})_{\pi(i)} = \sum_{j=i}^{n} t_j, \qquad i = 1, \dots, n, \qquad \mathbf{t} \ge 0. \tag{33}$$

This map is a bijection from $\mathbb{R}^n_{\geq 0}$ onto C_π, with inverse

$$t_i = x_{\pi(i)} - x_{\pi(i+1)} \quad (i < n), \qquad t_n = x_{\pi(n)}. \tag{34}$$

Consequently, a polynomial q is nonnegative on C_π if and only if $q \circ T_\pi$ is nonnegative on $\mathbb{R}^n_{\geq 0}$. The ordering domains may overlap when coordinates are equal, and their union still covers the entire domain.

Recursive certificate construction. The root contains the input polynomial p. At a node with polynomial $q(\mathbf{u})$, we expand it exactly and test whether

$$q(\mathbf{u}) = \sum_\alpha a_\alpha \mathbf{u}^\alpha, \qquad a_\alpha \geq 0 \text{ for every } \alpha. \tag{35}$$

Since every monomial is nonnegative for $\mathbf{u} \geq 0$, this condition certifies the node as a leaf. If the test fails and the search budget permits, we enumerate all orderings of the current variables and form the child polynomials $q_\pi(\mathbf{t}) = q(T_\pi \mathbf{t})$. Each child passing the coefficient test is closed; unresolved children are processed recursively by ordering and substituting their own nonnegative variables. An internal node is certified only when every ordering branch has been certified.

A successful certificate is a finite tree recording the ordering cases and substitution identities at internal nodes, together with the exact polynomial expansions and nonnegative coefficients at the leaves. The substitutions establish nonnegativity on each ordering domain, and coverage of these domains establishes nonnegativity at the parent. Applying this argument recursively proves $p \geq 0$ on $\mathbb{R}^n_{\geq 0}$. All substitutions, expansions, and coefficient checks use exact arithmetic. If the shared search budget is exhausted while any branch remains unresolved, the backend returns an unsolved result.

B.4 Lean Proof Generation and Verification

The Translator converts exact certificates from the three backends into local Lean proofs of the supplied polynomial statements. For SOS, it instantiates the recovered rational weighted-square identity, proves nonnegativity of the weights and squared terms, and applies the inequality and equality assumptions to the multiplier terms. For Bernstein certificates, it formalizes the affine substitutions, derives the unit-box bounds, and verifies the nonnegative basis expansion on each leaf box. It then combines these local results using the subdivision tree and coverage of the parent boxes. For Diff-Sub, it constructs the ordering cases, proves nonnegativity of the difference variables, and verifies the substitution identities and coefficient nonnegativity at the leaves. Polynomial identities and rational arithmetic are verified within Lean. A local goal is discharged only after the generated proof is accepted by the Lean kernel.

The agent incorporates the verified local result into the surrounding proof context and establishes any conditions connecting it to the original goal. These include the assumptions required by the local certificate and the side conditions of any reformulation, such as denominator positivity or variable-substitution identities. The resulting complete proof is checked against the original theorem statement, with its original assumptions, by the Lean kernel.

C Sources and Construction of LitIneq

We construct LitIneq by collecting 78 examples from research papers and formal-verification projects and formulating them as inequality-proving tasks in Lean 4. The benchmark combines polynomial bound-verification problems from global optimization with nonlinear inequalities from geometric and program verification.

Source collection. Table 6 summarizes the composition. We collect 45 polynomial instances from studies on Bernstein polynomials, global optimization, and inequality verification (Muñoz & Narkawicz, 2013; Liu et al., 2025; Ray & Nataraj, 2009), together with examples from the Flyspeck nonlinear-verification documentation. [1] For the APPIRL instances, we retain the benchmark formulations, problem names, and source attributions.

[1] *A Tool for Formal Verification of Nonlinear Inequalities*, Alexey Solovyev, Flyspeck project documentation.

Table 6: Sources of the 78 LitIneq instances. Counts refer to the collected benchmark statements, including source-derived variants.

Source group	Count	Instance identification
Bernstein examples	2	Coefficient and subdivision examples
APPIRL benchmark	10	C1–C10
Polynomial bounds from verification examples	8	Reaction Diffusion, Schwefel, Lotka–Volterra, Magnetism, Caprasse, Flyspeck Delta, Butcher, and Heart Dipole
Bernstein range-computation benchmark	25	A01–A18 and B01–B07
Flyspeck (HOL Light)	11	Original example identifiers
NASALib (PVS)	22	Original lemma names
Total	78	

The remaining 33 instances come from Flyspeck's HOL Light development[2] and NASA's PVS library, NASALib.[3]

Task construction and Lean formalization. For the 45 polynomial instances, we formulate bound-verification problems over explicit box domains. A selected lower bound $p(\mathbf{x}) \geq \ell$ is expressed as $p(\mathbf{x}) - \ell \geq 0$, with upper bounds handled analogously. Finite decimal coefficients and bounds are represented as exact rational numbers. For the HOL Light and PVS examples, we translate the selected statements and required definitions into Lean 4, making domain restrictions and positivity assumptions explicit. Each instance is packaged as a separate theorem with its own assumptions and target, retaining the strict or non-strict relation of the selected statement. Source identifiers and statement transformations are recorded for each instance.

Some instances involve explicit choices of domain or target. For A14 (Magnetism), we specify the box $[-5, 5]^7$ to resolve the ambiguous domain specification in the source. For B02 (Six-hump camel), we construct the target by subtracting a decimal approximation of the reported minimum from the objective polynomial. For B05 (Quadratic), we take $\sum_i x_i^2$ as the nonnegative target. For Flyspeck's Delta bound, the 45-instance bound-verification collection contains a non-strict version, while the HOL Light collection retains the strict version.

Polynomial instances. Of the 78 instances, 53 have targets and assumptions consisting of polynomial comparisons with rational coefficients. These comprise the 45 polynomial instances described above, five Flyspeck instances, and three NASALib instances. Table 7 summarizes their distribution by the number of quantified real variables and the total degree of the target polynomial. Fifty instances have box constraints, with two bound inequalities per variable and no explicit equality constraints. Among the remaining three, `simple` is unconstrained, `quadratic` has two polynomial inequality assumptions, and `ajn` has two alternative branches, each containing one equality and eleven inequalities.

Instances involving non-polynomial expressions. The remaining 25 instances comprise six Flyspeck instances and nineteen NASALib instances. They involve rational expressions with variable-dependent denominators, absolute values, square roots, logarithms, exponentials, and trigonometric or inverse trigonometric functions, including sine, tangent, and arctangent. Among them, 21 statements contain between one and eight quantified real variables, counting parameters as variables, while four are closed numerical inequalities. Non-polynomial expressions may occur in the target, the assumptions, or the constants defining the domain. The classification follows the input expressions before proof-oriented reformulations.

[2] `https://github.com/flyspeck/flyspeck`
[3] `https://github.com/nasa/pvslib`

Table 7: Distribution of the 53 polynomial instances by the number of variables n and target total degree d_p. Entries are instance counts. We count all quantified real variables, including parameters, without eliminating fixed variables. The degree is computed after expanding auxiliary definitions, simplifying rational constants, and moving the target to one side.

	Target total degree d_p			
Variables n	0–2	3–4	5–8	Total
1–2	6	2	2	10
3–4	4	11	0	15
5–6	8	8	3	19
7–8	4	2	2	8
11	0	1	0	1
Total	22	24	7	53

D Evaluation Protocol

All methods are evaluated with a two-hour time limit per problem. We use Lean and Mathlib v4.30.0 for all Lean-based methods. Each task retains its original assumptions and target statement; a successful Lean proof must establish that statement without `sorry` or `admit`.

Hardware and agent runtime. The local evaluation platform uses an Intel Core i9-10900K CPU at 3.70 GHz with 64 GB of DDR4 memory. IneqAgent and Direct Agent use `codex-cli` version `0.149.0` as the underlying LLM coding agent. API-based LLM inference is served externally; the locally deployed prover models use the GPUs specified below.

Symbolic computation. For Z3, we encode the assumptions together with the negated target as an SMT satisfiability query. An `unsat` result establishes the inequality, whereas `sat` indicates a counterexample; an `unknown` result or timeout is not counted as a success. For Maple, we use `PolynomialRing` from the `RegularChains` package to specify the polynomial ring and invoke `SamplePoints` to determine whether the corresponding counterexample constraints have real solutions. A problem is counted as solved when the computation establishes that no such solution exists. For Isabelle/HOL, we apply the `sos` method and count a success when Isabelle completes the formal proof.

LLM provers. We evaluate Goedel-Prover-V2-8B and DeepSeek-Prover-V2-7B at pass@32. A problem is solved if at least one of up to 32 proof attempts produces a Lean-verified proof within the two-hour limit. Both models are deployed for inference on NVIDIA L40S GPUs.

Agentic frameworks. For the published agentic baselines, we use the configurations associated with the best reported results in their respective papers, subject to the common two-hour limit. Our framework uses DeepSeek-V4-Flash for the main comparison in Table 1. Agentic methods have no additional caps on proof attempts, LLM calls, or token usage, and API inference is provided by external services. In the cost comparison in Table 2, our framework and Direct Agent use the same backbone in each pair. Direct Agent runs with its default configuration and access to Lean.

We report the percentage of problems solved and the mean time in seconds over successful problems. We compute overall success rate over all 190 problems and overall mean time over all problems successfully solved by each method across the four datasets.

LLM inference pricing. Table 8 lists the token rates used for the four backbones. Uncached input, cached input, and output tokens are charged at their respective rates, and inference cost is accumulated over all LLM calls in a run. Each IneqAgent–Direct Agent pair uses the same backbone and rates. Rates are reported in USD per million tokens and the cost comparisons in Section 4.3 are reported in US cents.

Table 8: LLM inference rates in USD per million tokens. Input denotes uncached input tokens.

Backbone	Input	Cached input	Output
DeepSeek-V4-Flash	0.2206	0.0074	0.6618
Qwen3.8-Flash	0.1176	0.0147	0.3971
Kimi-K3	2.9412	0.2941	14.7059
GPT-5.5	5.0000	0.5000	30.0000

E Details of the Dittert, Holens–Djokovic, and Merris Conjectures

We give the historical context of the conjectures in Section 4.5, the Dittert certificate constructions, and the proof of our four-dimensional Merris result. Throughout, J_n denotes the matrix whose entries are $1/n$, Ω_n is the set of doubly stochastic $n \times n$ matrices, and $A(i \mid j)$ is obtained by deleting row i and column j. We use $A \geq 0$ for entrywise nonnegativity and $Q \succeq 0$ for positive semidefiniteness of a symmetric matrix.

Table 9 shows that Dittert $n = 5$ has the largest expanded target, while HD $(4, 4)$ incurs the highest time and cost.

Table 9: Conjecture task sizes and costs. Dittert sizes refer to P_4 and $\widetilde{F}_5$ before constraint elimination. For HD $(4, 4)$, sizes refer to the reduced Holens–Djokovic polynomial Q_4, using the upper-left 3×3 entries as nine independent variables. Time is wall-clock time; Cost is total LLM inference cost.

Task	Variables	Total degree	Monomials	Time (s)	Cost (US cents)
Dittert, $n = 4$	16	4	521	5,292	84.6728
Dittert, $n = 5$	25	5	118,755	6,704	100.1654
HD $(4, 4)$	9	4	222	42,731	521.9610

E.1 Dittert's Conjecture

Let $K_n = \{A \in \mathbb{R}^{n\times n}_{\geq 0} : \sum_{i,j} a_{ij} = n\}$, and let r_i and c_j be the row and column sums of A. Dittert's conjecture, recorded in Minc's 1983 survey and restated as Conjecture 28 in the survey of Cheon and Wanless (Minc, 1983; Cheon & Wanless, 2005), asserts

$$\Phi_n(A) := \prod_{i=1}^{n} r_i + \prod_{j=1}^{n} c_j - \operatorname{per}(A) \leq 2 - \frac{n!}{n^n} \qquad (A \in K_n), \tag{36}$$

with equality exactly at J_n. On Ω_n, both products of sums equal one, so the inequality reduces to the van der Waerden lower bound $\operatorname{per}(A) \geq n!/n^n$. The additional difficulty is to control matrices in the larger simplex K_n, where neither the row sums nor the column sums are fixed individually.

Sinkhorn proved the $n = 2$ case and showed that a maximizing matrix has positive permanent (Sinkhorn, 1984). Hwang established that any entrywise positive maximizer must be J_n and that J_n is a strict local maximum (Hwang, 1986); he subsequently proved the $n = 3$ case (Hwang, 1987). These results leave the exclusion of nonuniform boundary maximizers as a central obstacle. Cheon and Wanless excluded partly decomposable matrices and matrices whose entire zero set forms a single block, and derived quantitative restrictions on the row and column sums of a possible counterexample (Cheon & Wanless, 2012). Udayan and Somasundaram also established full indecomposability of a maximizer in dimension four (Udayan & Somasundaram, 2024). These structural conditions alone do not identify the maximizer in that dimension. More recently, a 2026 preprint of Pang reports a proof for all $n \geq 17$ using refined scaling estimates (Pang, 2026). Our constructions below address dimensions four and five by explicit rational nonnegativity certificates. Both results include quantitative stability and uniqueness of the maximizer.

E.1.1 Dimension Four: A Quantitative Stability Certificate

Polynomial target and certificate template. Flatten the entries of A into $x = (x_1, \dots, x_{16})$, and set

$$h(x) = \sum_{i=1}^{16} x_i - 4, \qquad \Delta_4(x) = \sum_{i=1}^{16} (x_i - \tfrac{1}{4})^2, \qquad P_{4,\lambda}(x) = 61 - 32\Phi_4(A(x)) - \lambda\Delta_4(x). \tag{37}$$

It suffices to prove $P_{4,\lambda} \geq 0$ under $x_i \geq 0$ and $h = 0$ for one rational $\lambda > 0$. We search for a degree-four constrained SOS identity of the form

$$P_{4,\lambda} = \sigma_0 + \sum_{i=1}^{16} x_i\sigma_i + \sum_{1\leq i<j\leq 16} x_i x_j \sigma_{ij} + hq, \tag{38}$$

where σ_0 is SOS of degree at most four, σ_i and σ_{ij} are SOS of degree at most two, and $\deg q \leq 3$. Every term except hq is nonnegative on the domain, and hq vanishes there. The initial template using only $x_i\sigma_i$ does not yield a positive stability parameter in the numerical search. Adding the pairwise generators $x_i x_j$ yields a numerical value near 0.6153852, which motivates the rational candidate $\lambda = 8/13$. This value is accepted only after exact certification; we do not claim that it is the optimal stability parameter.

The resulting template has 136 small Gram blocks, corresponding to $16 + \binom{16}{2}$ generators, and one main block. The small blocks use $z_{\mathrm{sm}} = (1, x_1, \dots, x_{16})^\top$ and have size 17×17; the main block uses all monomials of degree at most two and has size 153×153. A numerical semidefinite solve supplies candidate matrices. Turning them into exact certificates requires handling both singularity and the coupling between blocks.

Removing the forced singular direction. Write $c = (1/4, \dots, 1/4)$. Since $P_{4,8/13}(c) = h(c) = 0$ and every generator $x_i, x_i x_j$ is positive at c, each SOS block must vanish at c. Thus, for a block $\sigma_b = z_b^\top G_b z_b$ with $G_b \succeq 0$, we have $G_b z_b(c) = 0$. Ordering z_b with its constant monomial first, write $z_b(c) = (1, \widetilde{z}_b(c)^\top)^\top$ and define

$$R_b = \begin{pmatrix} -\widetilde{z}_b(c)^\top \\ I \end{pmatrix}, \qquad G_b = R_b S_b R_b^\top. \tag{39}$$

This parameterization enforces the known kernel exactly. The numerical reduced matrices S_b are positive definite, making rational recovery possible without perturbing the forced zero direction.

Sequential rational recovery. We recover and fix the 136 small reduced blocks first. Their entries are rounded to rational values with a common candidate denominator, and each block is accepted only when an exact $LDL^\top$ factorization has positive diagonal entries. The successful common denominator in the construction is 1331. Once these blocks are fixed, let

$$C_{\mathrm{sm}} = \sum_i x_i\sigma_i + \sum_{i<j} x_i x_j \sigma_{ij}, \qquad \mathcal{R} = P_{4,8/13} - C_{\mathrm{sm}}.$$

The main block is rationalized and corrected to satisfy two conditions: its reduced matrix remains positive definite, and its residual vanishes identically after imposing the simplex equality. Explicitly, with $\sigma_0 = z_0^\top R_0 S_0 R_0^\top z_0$, where R_0 denotes the matrix R_b for the main block, we check

$$S_0 \succ 0, \qquad (\mathcal{R} - \sigma_0)\left(x_1, \dots, x_{15}, 4 - \sum_{i=1}^{15} x_i\right) \equiv 0. \tag{40}$$

The successful denominator parameter for the main-block recovery is 32767. Exact polynomial division then yields $q = (\mathcal{R} - \sigma_0)/h$ with zero remainder, establishing Eq. 38 coefficientwise over $\mathbb{Q}$.

Scaling and the exact code target. The normalized target above and the integer-scaled certificate represent the same inequality. Define

$$Q_{4,\mathrm{int}} = 61 - 32\Phi_4(A) = 32\big(61/32 - \Phi_4(A)\big), \qquad D_4^s = \sum_{i=1}^{16} x_i^2 - 1. \tag{41}$$

With our convention $h = \sum_i x_i - 4$, the centered distance satisfies $\Delta_4 = D_4^s - h/2$. Consequently,

$$\begin{aligned} F_s &:= 13Q_{4,\mathrm{int}} - 8D_4^s, \\ F_c &:= 13Q_{4,\mathrm{int}} - 8\Delta_4 = F_s + 4h = 13P_{4,8/13} \\ &= 416\left(\frac{61}{32} - \Phi_4(A) - \frac{1}{52}\Delta_4\right). \end{aligned} \tag{42}$$

The exported SOS identity for F_s has the same block structure as Eq. 38, with the weights scaled by 13. If its equality multiplier is q_{json}, the identity for F_c uses $q_{\mathrm{json}} + 4$ and the same SOS blocks. The Lean definition `compactTarget` is F_c; dividing its nonnegativity statement by 416 gives the stability constant $1/52$. The coefficient $8/13$ belongs to the 32-fold scaled gap in P_4; it must be divided by 32 exactly once to obtain $1/52$.

Conclusion and formal verification. The final certificate contains 152 positively weighted rational squares in the main SOS and 16 in each of the 136 small SOS blocks. The Lean formalization checks the rational weights, the nonnegativity of the constraint factors, and the polynomial identity, and relates the flattened polynomial to the matrix statement. Therefore

$$\frac{61}{32} - \Phi_4(A) \geq \frac{1}{52}\|A - J_4\|_F^2 \qquad (A \in K_4). \tag{43}$$

If $\Phi_4(A) = 61/32$, the squared norm must vanish, so $A = J_4$. Conversely, direct evaluation at J_4 gives equality. This establishes both the conjectured upper bound and the unique equality case. The updated proof record identifies `DH4_one_div_52_sos` as the coordinate-level stability theorem and lists matrix theorems for quantitative stability and the full four-dimensional conjecture in `DH4DittertOneDiv52Fast.lean`.

E.1.2 Dimension Five: Symmetry Reduction and Exact Recovery

Homogenization, distance term, and symmetry. For an arbitrary nonnegative 5×5 matrix, put $s = \sum_{i,j} a_{ij}$ and define

$$\begin{aligned} H_5(A) &= 1226s^5 - 5^9\Phi_5(A), \\ G_5(A) &= s^3 \sum_{i,j=1}^{5} (25a_{ij} - s)^2 = 625s^3 \sum_{i,j=1}^{5} a_{ij}^2 - 25s^5, \\ \widetilde{F}_5(A) &= 100H_5(A) - G_5(A). \end{aligned} \tag{44}$$

The strengthened certificate matches $\widetilde{F}_5$, rather than H_5 alone. It is a homogeneous degree-five polynomial in 25 variables with 118,755 monomials after expansion and collection. The factor s^3 makes the squared-distance term homogeneous of degree five as well. Independent row and column permutations, together with transposition, form a symmetry group $\mathcal{T}$ of order $2(5!)^2 = 28{,}800$. Its action partitions the degree-five monomials into 54 orbits. Let $v(A)$ list the $\binom{26}{2} = 325$ degree-two monomials in the entries. We seek a rational symmetric matrix $Q \succeq 0$ such that

$$\widetilde{F}_5(A) = \mathcal{S}_Q(A) := \sum_{\tau \in \mathcal{T}} \tau\big(a_{11}\, v(A)^\top Q v(A)\big), \tag{45}$$

where τ acts by permuting matrix-entry variables. The factor a_{11} raises the degree from four to five and is nonnegative on the domain. Since both sides are invariant, coefficient matching requires only one equation per monomial orbit, giving 54 linear equations for Q.

Enforcing the equality-point kernel. A numerical semidefinite feasibility solve gives an approximate Gram matrix. At the all-ones matrix, $H_5 = G_5 = \widetilde{F}_5 = 0$ and $v = \mathbf{1}_{325}$; positive semidefiniteness therefore forces $Q\mathbf{1}_{325} = 0$. To preserve this condition throughout rational recovery, write $u = \mathbf{1}_{324}$ and parameterize

$$Q(S) = \begin{pmatrix} S & -Su \\ -u^\top S & u^\top Su \end{pmatrix} = R^\top S R, \qquad R = (I_{324} \;\; -u). \tag{46}$$

Here S is a symmetric 324×324 matrix. This parameterization enforces symmetry and the zero direction identically, and $Q(S) \succeq 0$ if and only if $S \succeq 0$.

Exact coefficient correction. Vectorize the upper triangle of S into $t \in \mathbb{Q}^{52,650}$. The orbit equations become a rational linear system $Bt = b$, where b contains the orbit coefficients of $100H_5 - G_5$. Strengthening the target changes this right-hand side; the symmetry operator and kernel parametrization are unchanged. After rounding the numerical reduced matrix to a rational candidate t_{rat}, the residual is $b - Bt_{\mathrm{rat}}$. The completed matrix always has the zero direction, so evaluation at the all-ones matrix gives one dependency among the 54 equations. The construction identifies 53 independent rows and verifies their independence by exact elimination. Writing the selected system as $B_* t = b_*$, an exact correction is

$$\delta t = B_*^\top (B_* B_*^\top)^{-1}(b_* - B_* t_{\mathrm{rat}}), \qquad t_{\mathrm{exact}} = t_{\mathrm{rat}} + \delta t. \tag{47}$$

This is the minimum Euclidean-norm correction in the chosen upper-triangle coordinates. It limits the perturbation of the numerical candidate but does not itself prove positive semidefiniteness. All 54 original equations are checked again in exact arithmetic after the correction.

Exact positivity and quantitative stability. Positive semidefiniteness must be checked in exact arithmetic after coefficient correction. An exact $LDL^\top$ decomposition of the reduced matrix with nonnegative diagonal weights suffices: by Eq. 46, it gives a rationally weighted sum-of-squares representation of $v^\top Q v$. The strengthened certificate is accepted only after checking symmetry, the exact kernel equation, positive semidefiniteness, and all coefficient identities for $100H_5 - G_5$. Each summand in Eq. 45 is consequently nonnegative, proving $100H_5(A) - G_5(A) \geq 0$ for every $A \geq 0$, without a normalization assumption. At $s = 5$, let $\Delta_5 = \|A - J_5\|_F^2$. Then

$$G_5 = 5^7 \Delta_5, \qquad \frac{\widetilde{F}_5}{100 \cdot 5^9} = \frac{1226}{625} - \Phi_5(A) - \frac{1}{2500}\Delta_5. \tag{48}$$

Thus the normalized matrix conclusion is

$$\frac{1226}{625} - \Phi_5(A) \geq \frac{1}{2500}\|A - J_5\|_F^2 \qquad (A \in K_5). \tag{49}$$

If $\Phi_5(A) = 1226/625$, the positive stability coefficient forces $\Delta_5 = 0$, hence $A = J_5$. Direct substitution gives equality at J_5. This establishes the conjectured upper bound and uniqueness, including on the boundary of K_5. Uniqueness follows from the distance estimate; it does not require the strengthened remainder itself to have only one zero.

Lean verification and the matrix theorem. The updated proof record identifies `h5Poly` with H_5, `uniquenessGapPoly` with G_5, and `compactTargetPoly` with $100H_5 - G_5$. The theorem `eval_compactTargetPoly_nonnegative` checks the strengthened target on the nonnegative orthant; `DH5_strengthened` then states $G_5/100 \leq H_5$. The matrix-level file `DH5Main.lean` applies $s = 5$ and Eq. 48 to obtain the $1/2500$ stability bound and the unique equality case. These are the quantitative-stability and dimension-five conjecture theorems listed in the updated record.

E.2 Holens–Djokovic (4,4) and Merris's Conjecture

Merris proposed in 1973 that every $A \in \Omega_n$ satisfies

$$\min_{1 \leq j \leq n} S_j(A) \leq n \operatorname{per}(A), \qquad S_j(A) := \sum_{i=1}^{n} \operatorname{per}(A(i \mid j)) \tag{50}$$

(Merris, 1973). The permanent has the weighted column expansion $\operatorname{per}(A) = \sum_i a_{ij} \operatorname{per}(A(i \mid j))$. The conjecture asks for at least one column whose corresponding *unweighted* sum of minors is at most n times the permanent; this does not follow from the expansion alone.

A related result of Marcus and Merris gives $H^{1/2} \operatorname{Padj}(H) H^{1/2} \preceq n \operatorname{per}(H) I_n$ for positive semidefinite Hermitian H, where $\operatorname{Padj}(H)_{ji} = \operatorname{per}(H(i \mid j))$. For positive semidefinite doubly stochastic matrices, their averaged minor inequality implies Eq. 50 (Marcus & Merris, 1973). General doubly stochastic matrices need not be symmetric or positive semidefinite, so this argument does not cover the conjecture in full. Udayan and Somasundaram's 2021 paper records the general problem as open for $n \geq 4$ and proves it for additional classes, including matrices constant within each column whose largest entry is at least $1/n$ (Udayan & Somasundaram, 2021).

E.2.1 Dimension Four: An SOS Proof via Holens–Djokovic

Merris target and the certified polynomial. For $A \in \Omega_4$, write

$$S_j(A) = \sum_{i=1}^{4} \operatorname{per}(A(i \mid j)), \qquad M_{4,j}(A) = 4\operatorname{per}(A) - S_j(A). \tag{51}$$

The desired conclusion is

$$\forall A \in \Omega_4, \qquad \exists j \in \{1,2,3,4\}, \quad M_{4,j}(A) \geq 0. \tag{52}$$

The SOS certificate directly proves the $(4,4)$ instance of the Holens–Djokovic conjecture (Holens, 1964; Djoković, 1967) in the polynomial form

$$Q_4(A) := 24\operatorname{per}(A) - 2 - \sum_{i,j=1}^{4} a_{ij}^3 \geq 0, \qquad A \in \Omega_4. \tag{53}$$

As shown below, this implies a bound on the average of the four S_j, which in turn gives Eq. equation 52. Thus Q_4 is a sufficient polynomial target for the Merris result; we do not identify it with any individual $M_{4,j}$.

Nine-variable parametrization. Let $u = (u_{11}, u_{12}, \ldots, u_{33})$ contain the upper-left 3×3 entries. Define $A(u)$ by

$$\begin{aligned} a_{ij} &= u_{ij} && (i,j \leq 3), \\ a_{i4} &= 1 - \sum_{j=1}^{3} u_{ij}, \qquad a_{4j} = 1 - \sum_{i=1}^{3} u_{ij} && (i,j \leq 3), \\ a_{44} &= \sum_{i,j=1}^{3} u_{ij} - 2. \end{aligned} \tag{54}$$

All row and column sums are then one. Every matrix in Ω_4 has this representation, and its feasible domain is exactly the sixteen linear inequalities $a_{ij}(u) \geq 0$. Consequently, the parametrization eliminates the equality constraints without relaxing nonnegativity. The polynomial $\widehat{Q}_4(u) = Q_4(A(u))$ has nine variables, total degree four, and 222 nonzero monomials after expansion and collection. These are the sizes reported for the Merris experiment in Section 4.5; 222 counts the terms of this single SOS target.

Symmetrized rational SOS certificate. Let G consist of the row permutations, column permutations, and transposition, with $|G| = 2(4!)^2 = 1152$. These transformations preserve Ω_4 and Q_4. The certificate uses four representative multipliers,

$$m_0(A) = a_{41}a_{42}a_{43}a_{44}, \quad m_1(A) = a_{44}, \quad m_2(A) = a_{44}a_{43}, \quad m_3(A) = a_{44}a_{33}, \tag{55}$$

and has the form

$$\widehat{Q}_4(u) = \sum_{r=0}^{3} \sum_{\tau \in G} \sum_{k=1}^{N_r} \lambda_{r,k}\, m_r(\tau A(u))\, q_{r,k}(\tau A(u))^2, \qquad \lambda_{r,k} \in \mathbb{Q}_{\geq 0}. \tag{56}$$

Here each $q_{r,k}$ has rational coefficients; when expressed in nine coordinates, its argument is the upper-left block of $\tau A(u)$. The four blocks have $(N_0, N_1, N_2, N_3) = (9, 24, 42, 34)$ weighted squares before summing over transformations. The implementation partitions the blocks into $(8, 16, 24, 24)$ slices, respectively, for a total of 72 slices. This partition permits parallel checking of the same algebraic identity. For every feasible u, all transformed matrix entries, multipliers, weights, and squares are nonnegative. Hence the certificate establishes Eq. 53, including matrices with zero entries.

The certificate is discovered by semidefinite programming on the nine-variable model. Known equality cases, including the uniform matrix, impose vanishing conditions that reduce the Gram-block dimensions. Numerical Gram data are recovered over the rationals and factored into weighted squares. The exported weights, polynomials, symmetry mappings, and target coefficients are then checked by exact polynomial expansion. The numerical search supplies a candidate; acceptance depends on the rational identity and nonnegativity checks.

From Holens–Djokovic to Merris. For doubly stochastic 4×4 matrices, the proof uses the identity

$$3\sum_{j=1}^{4} S_j(A) = 4 + 2\sum_{i,j=1}^{4} a_{ij}^3. \tag{57}$$

After substitution of Eq. 54, both sides are polynomials in u, and the identity is checked by exact expansion. Combining it with the SOS certificate gives

$$16\,\mathrm{per}(A) - \sum_{j=1}^{4} S_j(A) = \frac{2}{3} Q_4(A) \geq 0. \tag{58}$$

Choose a column j_* minimizing $S_j(A)$. Then

$$S_{j_*}(A) \leq \frac{1}{4}\sum_{j=1}^{4} S_j(A) \leq 4\,\mathrm{per}(A), \tag{59}$$

which proves Eq. 52. This argument applies directly to every $A \in \Omega_4$ and does not require a limiting argument.

Formal verification and the public theorem's scope. The supplied proof record describes the `HolensDjokovic44Fast` project using Lean and Mathlib version `v4.30.0-rc1`. Its sparse-polynomial modules establish evaluation semantics, while the symmetry and multiplier lemmas justify nonnegativity of each orbit certificate. Each slice checks rational weight nonnegativity and an exact polynomial identity. The certificate module combines all four blocks and identifies their sum with $24\,\mathrm{per}(A(u)) - 2 - \sum_{i,j} a_{ij}(u)^3$. The final matrix theorem connects the parametrization to Mathlib's `Matrix.permanent` for arbitrary $A \in \Omega_4$.

The public theorem `holens_djokovic_4_4` states Eq. 53. The public theorem `merris_n4` assumes, in addition to $A \in \Omega_4$, that

$$S_4(A) \leq S_1(A), \qquad S_4(A) \leq S_2(A), \qquad S_4(A) \leq S_3(A),$$

and concludes $S_4(A) \leq 4\,\mathrm{per}(A)$. For the mathematical existential statement, any minimizing column can be moved to position four by a column permutation. Such a permutation preserves double stochasticity and the permanent, and permutes the values S_j. Applying the stated theorem to the permuted matrix and relabeling the column yields Eq. 52. The supplied record documents the fixed-minimum-column Lean theorem; it does not list a separate public theorem with the existential conclusion.

The record reports no `sorry`, `admit`, or project-defined axioms. Large finite equalities are checked with `native_decide`, whose trusted base includes Lean's native code generator and runtime.