AI-Assisted Discovery of a Polynomial Lyapunov Potential for the Antiferromagnetic Potts Model

Chendong Song [1] **Chenlin Gu** [2] **Zijie Zhou** [1]

Abstract

Progress in AI for mathematics is largely measured on benchmarks and competition problems, leaving a gap with the questions working mathematicians actually care about. We argue for a different entry point: problems whose key idea is simple once seen, yet whose search and verification are tedious. As a case study from a working mathematician's perspective, we revisit an already-resolved problem and find a genuinely new proof of it: uniqueness of the Gibbs measure for the three-state antiferromagnetic Potts model on the infinite binary tree at every positive temperature. Where prior proofs control the tree recursion through two-step compositions and delicate computer-assisted estimates, we construct a new one-step polynomial Lyapunov potential on the simplex that reduces the whole problem to a single finite-dimensional inequality—which we then discharge by harnessing AI's strength in symbolic computation and exact verification, via a Bernstein-basis certificate. Beyond the proof itself, we report on the agentic workflow that led to it: we tried several AI-driven paths to this problem—autonomous proof search, adversarial potential discovery, and formal-prover attempts—and summarize what worked and what did not, offering a few concrete lessons for using current agents in this kind of search-and-verify mathematics.

1. Introduction

In recent years the mathematical reasoning abilities of large language models (LLMs) have advanced rapidly, progressing from elementary arithmetic to competition, graduate, and research-level mathematics. Early models such as GPT-3 struggled with basic arithmetic, whereas GPT-4 scored 92% on the grade-school benchmark GSM8K (Cobbe et al., 2021); with test-time scaling, systems such as OpenAI's o1 went on to perform strongly on high-school competitions including AIME, and Google's Gemini Deep Think reached gold-medal–level performance at IMO 2025 using purely natural-language reasoning (Google DeepMind, 2025). The frontier has continued to climb: frontier models now exceed 90% on undergraduate mathematics exams and score above 80 on PhD qualifying exams (Glazer et al., 2024), and general-purpose models have begun to touch genuine research—an OpenAI model recently produced an AI-generated disproof in the classical extremal setting of Erdős's planar unit-distance problem, discovering a new family of constructions (Alon et al., 2026). In parallel, formal verification has emerged as a key driver of reliability: Lean 4 provides a machine-checkable language for definitions, statements, and proofs (Moura & Ullrich, 2021), and agentic systems that pair natural-language reasoning with a formal backend now push further still—Rethlas, coupling an informal reasoning agent with a formal verification agent, autonomously resolved an open problem in commutative algebra and formalized the proof in Lean 4 with essentially no human intervention (Ju et al., 2026).

Yet as capability advances toward the frontier, a persistent gap between the AI and mathematics communities comes into focus, rooted in how each side frames a problem. On the AI side, progress is largely measured through scalable benchmarks, formal environments, and competition-style problem sets. Such settings are valuable—they offer clear feedback and reproducible evaluation—but they tend to reflect the capability of the AI system rather than to engage problems that working mathematicians already regard as structurally important, and much of this work consequently sits at a distance from frontier pure mathematics. On the mathematics side, research problems are rarely handed to a machine as a clean pipeline: the difficulty usually lies in choosing the right invariant, identifying the correct normalization, or finding a proof object that makes a computation conceptually transparent. Before a theorem can be formalized or checked, one often has to discover the right

[1] Department of Industrial Engineering and Decision Analytics, HKUST, Hong Kong, China [2] Yau Mathematical Sciences Center, Tsinghua University, Beijing, China . Correspondence to: Chendong Song <songcd@ust.hk>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

mathematical object to formalize in the first place. This mismatch is telling: despite a number of outstanding agent systems (Ju et al., 2026; Liu et al., 2026), many working mathematicians still find it more natural to reason directly in dialogue with a general-purpose model such as GPT than to route their problem through a specialized pipeline.

Narrowing this gap, we suggest, is less a matter of building better pipelines than of choosing the right problems to bring to the machine in the first place. A productive entry point for AI-assisted mathematics, we argue, may not be the most conceptually difficult theorem, but a class of problems where the key idea is simple once seen while the search for it and the verification of candidates are tedious. The comparative advantage of AI over human mathematicians lies in search and computation, and the emergence of mathematical agents has sharpened this advantage further. AI assistance is therefore most valuable precisely where the bottleneck is not insight but the laborious process of generating, falsifying, and certifying candidates.

We present such a case study from the perspective of a working mathematician: the uniqueness problem for the three-state antiferromagnetic Potts model on the binary tree. The model is simple to define and structurally elegant, yet monotonicity inequalities are scarce on the Potts model, so prior proofs rely largely on technical, brute-force inequality estimates. Its clean level-to-level recursion has long made a recursive invariant—a Lyapunov function—the natural object to seek, but such an invariant is hard to find and verify by hand: the standard max/min constructions fail under one-step recursion and force one into a more elaborate two-step argument.

Contributions.

1. We formulate a new proof of uniqueness for the three-state antiferromagnetic Potts model on the binary tree ($q = 3$, $d = 2$) at every positive temperature, by constructing an explicit one-step Lyapunov potential—in contrast to the two-step recursions and computer-assisted estimates of prior work.

2. We systematically analyze the respective roles of AI and the human mathematician in exploring this problem, and examine the difficulties that current AI agents face along their exploration paths.

2. Mathematical Background

The Potts model and the uniqueness problem. The Potts model is a classical model in statistical physics for how local interactions produce large-scale order (Wu, 1982). One assigns a color from q states to each vertex of a graph and weights every configuration by how neighboring spins agree; in the *antiferromagnetic* regime, which we study, neighbors prefer to *differ*—the same flavor as proper graph coloring—with a temperature $w \in (0, 1)$ tuning this preference from forcing adjacent spins to differ ($w \to 0$) to no interaction at all ($w \to 1$). On infinite systems this gives rise to a *phase transition*: the influence of a far-away boundary may either persist deep inside the system or dissipate. The infinite $(d+1)$-regular tree is a canonical geometry for studying this, with tight links to counting and sampling in computer science, and the antiferromagnetic case is notoriously the hardest, since it lacks the monotonicity that in simpler models lets one bound the boundary's effect by sandwiching it between extreme cases (Galanis et al., 2018).

We now state the question precisely. Let $T_{d,n}$ be the rooted $(d+1)$-regular tree of height n, with leaf set Λ_n and root r_n; a *coloring* is a map $\sigma : V(T_{d,n}) \to [q]$. For temperature $w \in (0, 1)$ assign each coloring the weight

$$\mathrm{wt}(\sigma) = w^{m(\sigma)}, m(\sigma) = \#\{uv \in E(T_{d,n}) : \sigma(u) = \sigma(v)\}, \tag{1}$$

which penalizes every monochromatic edge by a factor $w < 1$. A *boundary condition* $\tau : \Lambda_n \to [q]$ fixes the colors of the leaves—the information imposed "far away"—and the finite-volume *Gibbs measure* $\mu^{\tau}_{n,w}$ is obtained by normalizing $\mathrm{wt}(\sigma)$ over all colorings consistent with τ. The quantity of interest is the law of the *root* color, $\mu^{\tau}_{n,w}(\sigma(r_n) = c)$ for $c \in [q]$. The infinite-tree model has a *unique* Gibbs measure exactly when the boundary is asymptotically forgotten: the root color becomes uniformly distributed regardless of the boundary,

$$\forall c \in [q], \quad \lim_{n\to\infty} \max_{\tau:\Lambda_n\to[q]} \left| \mu^{\tau}_{n,w}(\sigma(r_n) = c) - \tfrac{1}{q} \right| = 0. \tag{2}$$

Method recap. At its core, the uniqueness problem is a statement about a single map. The state of a subtree is summarized by the distribution of its root color, a vector in the probability simplex Δ_q over the q colors, and the distribution at a vertex is determined by those of its children through a one-step recursion Φ_w: if a vertex has d children whose subtree marginals are $x^{(1)}, \ldots, x^{(d)} \in \Delta_q$, its own marginal is $\Phi_w(x^{(1)}, \ldots, x^{(d)})$, which is defined for $i \in [q]$ by

$$\Phi_w(x^{(1)}, \ldots, x^{(d)})_i = \frac{\prod_{j=1}^{d} \left(1 - (1-w)x_i^{(j)}\right)}{\sum_{k=1}^{q} \prod_{j=1}^{d} \left(1 - (1-w)x_k^{(j)}\right)}. \tag{3}$$

Uniqueness is exactly the assertion that iterating Φ_w from any boundary drives the marginal to the uniform fixed point $u = (1/q, \ldots, 1/q)$. Every approach to the problem therefore reduces, in essence, to controlling the contraction of Φ_w on the simplex; the entire difficulty lies in proving that

this rational map pulls all inputs toward u at a uniform rate. Because Φ_w carries no monotonicity, none of the existing proofs controls it through a clean invariant; instead they proceed by technical, problem-specific inequality estimates. Galanis, Goldberg, and Yang analyze not Φ_w itself but its *two-step* composition $\Phi_w \circ \Phi_w$, bounding the resulting contraction by a sequence of explicit inequalities on the message coordinates, with parts of the case analysis discharged by rigorous but computer-assisted estimates that are not readily checked by hand (Galanis et al., 2018). De Boer, Buys, and Regts take a different but equally technical route, recasting the recursion in terms of complex dynamics and proving contraction via a zero-free region for an associated function, again through delicate analytic bounds (de Boer et al., 2023). In both cases the argument is driven by the manipulation of inequalities rather than by a single conceptual quantity: the proofs are rigorous but long, opaque, and tailored to the specific (q, d). What is conspicuously missing is a *one-step* potential—a single scalar function of the message that provably decreases under Φ_w—which would replace the two-step bookkeeping and case-by-case estimates with one inequality.

The potential-finding problem. This motivates the central object of our case study. We seek a *Lyapunov potential*: a scalar function $W : \Delta_q \to \mathbb{R}_{\geq 0}$ that vanishes only at the uniform vector u and that strictly decreases under the recursion, in the sense that

$$W\big(\Phi_w(x^{(1)}, \ldots, x^{(d)})\big) \;\leq\; \lambda \max_j W\big(x^{(j)}\big) \tag{4}$$

for some contraction factor $\lambda < 1$ and all inputs $x^{(1)}, \ldots, x^{(d)} \in \Delta_q$. A single such W would settle uniqueness in one step: iterating Equation (4) from the leaves forces W at the root to zero geometrically, hence the root distribution to u, for every boundary. The difficulty is not in this implication but in exhibiting the function itself. The most intuitive candidate—measuring the deviation from u by the largest or smallest coordinate, i.e. a $\max / \min$-type potential—is the one most previous analyses gravitate toward, but it simply does not contract under the one-step recursion: there are inputs for which a single application of Φ_w increases it. This failure is exactly what forces prior work into the more elaborate two-step recursion. Finding a potential that already works in one step is therefore the crux. The space of candidates is large, the constraint Equation (4) must hold uniformly over the entire product simplex Δ_q^d, and a candidate can fail on a thin region or only at high order, so that plausible guesses are easily but invisibly wrong. Verifying a proposed W amounts to certifying a single multivariate polynomial inequality over Δ_q^d—conceptually elementary, but tedious and error-prone to carry out by hand. This is precisely the kind of search-and-verify task where an AI agent can be brought to bear, and it is the problem we study in the rest of the paper.

3. The Polynomial Lyapunov Proof

We now specialize the potential-finding problem from Section 2 to the three-state model on the *rooted binary tree*: every internal vertex has exactly two children, so the one-step recursion Equation (3) specializes to the two-child map $\Phi_w(x, y)$ with $x, y \in \Delta_3$. (Equivalently, these are the directed-edge messages on the 3-regular tree, each vertex receiving two incoming child messages.) The proof is organized around one explicit scalar function. Let

$$u = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) \tag{5}$$

and define, for $p \in \Delta_3$,

$$W(p) := \sum_{i=1}^{3} \left(p_i - \frac{1}{3}\right)^2 - \frac{3}{4} \sum_{i=1}^{3} \left(p_i - \frac{1}{3}\right)^3. \tag{6}$$

Writing $a_i = p_i - 1/3$, this is

$$\begin{aligned} &W(p) = \sum_{i=1}^{3} a_i^2 \left(1 - \frac{3}{4} a_i\right), \\ &\sum_{i=1}^{3} a_i = 0, \\ &-\frac{1}{3} \leq a_i \leq \frac{2}{3}, \qquad \forall 1 \leq i \leq 3. \end{aligned} \tag{7}$$

Thus the cubic correction does not destroy positivity: $1 - (3/4)a_i \geq 1/2$ on the simplex.

Lemma 3.1 (Basic properties of the potential)**.** *For every $p \in \Delta_3$,*

$$W(p) \geq \frac{1}{2} \sum_{i=1}^{3} \left(p_i - \frac{1}{3}\right)^2. \tag{8}$$

Consequently $W(p) \geq 0$, with equality if and only if $p = u$. Moreover, W is convex on Δ_3.

The proof is elementary and is given in Section A.1. The only non-immediate part is convexity: although one diagonal entry of the ambient Hessian can be negative near a vertex, its restriction to the tangent plane $h_1 + h_2 + h_3 = 0$ is nonnegative.

Remark 3.2 (Why the cubic correction appears). The pure quadratic potential $Q(p) = \sum_i (p_i - 1/3)^2$ fails already on the diagonal family

$$x = y = p_\varepsilon = \left(\frac{1 - 2\varepsilon}{3}, \frac{1 + \varepsilon}{3}, \frac{1 + \varepsilon}{3}\right), \qquad 0 < \varepsilon \ll 1,$$

where

$$Q\big(\Phi_0(p_\varepsilon, p_\varepsilon)\big) - Q(p_\varepsilon) = \frac{1}{3}\varepsilon^3 + O(\varepsilon^4) > 0.$$

For the family $Q + \lambda \sum_i (p_i - 1/3)^3$, cancellation of the third-order obstruction forces $\lambda = -3/4$, giving Equation (6).

For the zero-temperature binary recursion we write $\Psi = \Phi_0$, i.e. for $x, y \in \Delta_3$,

$$\Psi(x,y)_i = \frac{(1-x_i)(1-y_i)}{\sum_{k=1}^{3}(1-x_k)(1-y_k)}, \qquad i \in [3]. \quad (9)$$

Equivalently, with $L_0(x)_i = (1-x_i)/2$, we have

$$\Psi(x,y) = \mathrm{Normalize}\big(L_0(x) \odot L_0(y)\big)$$

, where $\odot$ is the pointwise product. Since

$$S(x,y) := \sum_{k=1}^{3}(1-x_k)(1-y_k) = 1+\sum_{k=1}^{3} x_k y_k \geq 1, \quad (10)$$

the denominator is strictly positive, so $\Psi : \Delta_3 \times \Delta_3 \to \Delta_3$ is well defined. The main input to the proof is a sharp non-expansion inequality for Ψ.

Proposition 3.3 (Zero-temperature average non-expansion)**.** *For all $x, y \in \Delta_3$,*

$$W\big(\Psi(x,y)\big) \leq \frac{W(x)+W(y)}{2}. \quad (11)$$

Consequently,

$$W\big(\Psi(x,y)\big) \leq \max\{W(x), W(y)\}. \quad (12)$$

Proof sketch. Set

$$G(x,y) := \frac{W(x)+W(y)}{2} - W(\Psi(x,y)).$$

By Equation (10), $S(x,y) \geq 1 > 0$, so the inequality $G \geq 0$ is equivalent to

$$\mathcal{P}(x,y) := 8S(x,y)^3 G(x,y) \geq 0. \quad (13)$$

Using the symmetric identity

$$4W(p) = 2 - 5\sum_{1\leq i<j\leq 3} p_i p_j - 9p_1p_2p_3, \quad (14)$$

$\mathcal{P}$ becomes an explicit polynomial on $\Delta_3 \times \Delta_3$.

The certificate has two parts. Near the interior equality point we work on the box

$$C_0 = \left\{ p \in \Delta_3 : \frac{1}{4} \leq p_i \leq \frac{5}{12} \text{ for } i = 1,2,3 \right\}.$$

In real Fourier coordinates around u, the polynomial $6R^3G$ is expanded exactly and grouped according to the degree in the difference variable d:

$$K = K_0 + K_2 + K_{\geq 4}.$$

The diagonal term K_0 is nonnegative, the quadratic term K_2 dominates a positive multiple of $|d|^2$, and the higher-order remainder is bounded by a smaller multiple of $|d|^2$. This proves $G \geq 0$ on $C_0 \times C_0$.

On the complement of $C_0 \times C_0$, we use an exact Bernstein subdivision certificate. The simplex is triangulated by the mesh $1/6$ rational grid; cells near the centre are subdivided once, and the finitely many annular cells not certified at that scale are subdivided once more. On each product cell, the affine pullback of $9\mathcal{P}$ is expanded in the degree-six tensor-product Bernstein basis. All Bernstein coefficients are checked over $\mathbb{Q}$ and are nonnegative on the certified cells. Since Bernstein basis functions are nonnegative, this proves $\mathcal{P} \geq 0$ outside $C_0 \times C_0$. The detailed algebra, subdivision, and certificate counts are recorded in Sections A.3 and A.4; the verification scripts are listed in Section B. □

We now pass from zero temperature to positive temperature. Define the one-child channel

$$L_w(x)_i = \frac{1-(1-w)x_i}{2+w}. \quad (15)$$

Then

$$\Phi_w(x,y) = \mathrm{Normalize}\big(L_w(x) \odot L_w(y)\big), \quad (16)$$

and L_0 is the channel of Equation (9).

Lemma 3.4 (Uniform-noise decomposition)**.** *For $0 < w < 1$,*

$$L_w(x) = \alpha L_0(x) + \beta u, \qquad \alpha = \frac{2(1-w)}{2+w}, \qquad \beta = \frac{3w}{2+w}. \quad (17)$$

In particular, $\alpha, \beta > 0$ and $\alpha + \beta = 1$.

The identity follows by checking each coordinate. It says that positive temperature is obtained by mixing the zero-temperature child channel with uniform noise.

Theorem 3.5 (Strict Lyapunov contraction)**.** *For every $w \in (0,1)$ and all $x, y \in \Delta_3$,*

$$W\big(\Phi_w(x,y)\big) \leq \frac{2}{2+w}\max\{W(x), W(y)\}. \quad (18)$$

Proof sketch. Put $a = L_0(x)$, $b = L_0(y)$. By Theorem 3.4, $L_w(x) = \alpha a + \beta u$ and $L_w(y) = \alpha b + \beta u$. Expanding the pointwise product and normalizing expresses $\Phi_w(x,y)$ as a convex combination of

$$\Psi(x,y), \qquad \Psi(x,u), \qquad \Psi(u,y), \qquad u.$$

Convexity of W (Theorem 3.1), Theorem 3.3, and $W(u) = 0$ give a loss at least equal to the uniform component. More precisely, if $M = \max\{W(x), W(y)\}$, then

$$W(\Phi_w(x,y)) \leq \left(1 - \frac{\beta}{3}\right) M = \frac{2}{2+w} M.$$

The full weight calculation is in Section A.5. □

Theorem 3.6 (Uniqueness and exponential decay)**.** *For every $w \in (0,1)$, every color $c \in [3]$, and every height n,*

$$\max_{\tau:\Lambda_n \to [3]} \left| \mu^{\tau}_{n,w}(\sigma(r_n) = c) - \frac{1}{3} \right| \le \left(\frac{2}{2+w} \right)^{n/2}. \quad (19)$$

In particular, the Gibbs measure on the infinite binary tree is unique for every positive temperature $w \in (0,1)$.

Proof sketch. Let M_h be the largest possible value of W at the root of a height-h binary subtree, over all boundary conditions. Since leaves have delta messages and $W(\delta_i) = 1/2$, $M_0 \le 1/2$. By Theorem 3.5,

$$M_{h+1} \le \frac{2}{2+w} M_h, \qquad M_n \le \frac{1}{2} \left(\frac{2}{2+w} \right)^n.$$

The root message is the root marginal, and Equation (8) converts the decay of W into the coordinate bound Equation (19). The details are given in Section A.5. □

Remark 3.7 (Comparison with the sharp decay rate). For $q = 3$, $d = 2$ the critical temperature is $w_c = 1 - q/(d+1) = 0$, so the entire range $w \in (0,1)$ is subcritical. In this regime Gu, Wu, and Yang (Gu et al., 2025) determine the *sharp* exponential decay rate: the root deviation decays per level by the factor

$$\rho(w) = \frac{d(1-w)}{w+q-1} = \frac{2(1-w)}{2+w},$$

which is exactly the contraction coefficient α of Theorem 3.4. Our bound Equation (19) corresponds to a per-level factor $\sqrt{2/(2+w)} \ge \rho(w)$ on $(0,1)$, so the rate we obtain is not optimal; the slack comes from the single step $\alpha^2 \le 2/(2+w)$ in the proof of Theorem 3.5. The two results are nonetheless complementary in nature. The sharp-rate analysis of (Gu et al., 2025) *assumes* uniqueness (their Hypothesis 1.1) and only reads the rate off the behaviour of the recursion near its fixed point, i.e. it needs a local contraction estimate in a neighbourhood of u; for $q = 3, d = 2$ that hypothesis is supplied by (Galanis et al., 2018), and alternatively by the present work. Our Lyapunov inequality Equation (18), by contrast, holds on the entire product simplex $\Delta_3 \times \Delta_3$ rather than only near u, so a single global potential establishes uniqueness unconditionally and in one step—at the cost of a non-optimal constant, but removing both the uniqueness assumption and the two-step bookkeeping of prior approaches.

4. Agentic Assistance: A Case Study in Human–AI Collaboration

The proof above was obtained through an extended interaction between the authors and a general-purpose model (ChatGPT, GPT-5.5 Pro; accessed via the web interface in April–May 2026). The human role was not to carry out the large symbolic expansions by hand, but to supply the mathematical framing and the strategic moves; the model contributed breadth of candidate construction, symbolic labor, and exact verification. We do not present this as a standardized benchmark, but as a case study in where current mathematical agents helped, where they did not, and why the final proof still required human judgment. To keep the account auditable, representative transcripts and the verification artifacts are included in the supplementary material (Section C).

The process was genuinely interactive, and it is worth tracing step by step how the proof actually came about. At each stage the human supplied a strategic move—or one of two decisive insights—while the model supplied breadth of construction, symbolic computation, and exact verification. Table 1 records this division of labor across the project.

Remark 4.1 (Where the present challenge lies). What Table 1 highlights is that the decisive moves were largely strategic: choosing to look for a potential at all, finding the parameterization that made the search tractable, reducing positive temperature to zero temperature, and splitting the domain for the certificate. Each is a small but genuine act of mathematical judgment, and in our experience these high-level choices are the part of the workflow that current agents are least ready to take on by themselves—which is also what makes them the most interesting target. We see this less as a limitation than as a signpost: as agents become better at proposing such strategic steps, even partially, the human–AI loop in problems like this should tighten considerably. The rest of this section describes our attempts to delegate the individual steps to an agent, and what we learned from them.

4.1. Step 2: autonomous proof search

We first tried to delegate the whole of steps 1–5 at once. Following the design of recent formal-reasoning agents—specifically, we reproduced the agent loop of the Rethlas/Archon system (Ju et al., 2026) rather than calling any hosted service—we asked the agent to resolve the Potts-tree uniqueness problem directly, including exploratory search over related parameter regimes. We ran three such experiments; the longest lasted roughly 41 iterations and about ten hours, and none produced a proof.

The recurring difficulty was not a single incorrect calculation but a loss of global direction. Once the agent entered a branch of the search, it tended to keep refining that branch after it had stopped being relevant to the main obstruction; in one run it spent many iterations on a subcase of a two-step recursion, each step reporting plausible local progress and "one remaining lemma," without reassessing whether that lemma was central or had become harder than the original problem. A second difficulty was definition drift: across

	Human (insight & strategy)	AI (search, construction & verification)
1	Frame the problem through a Lyapunov potential—a long-standing "common sense" instinct among researchers for such tree recursions.	Propose concrete candidate forms for such a potential.
2	—	Attempt to construct a potential and a proof fully autonomously *(unsuccessful)*.
3	Curate and filter the candidate potentials; introduce the method of undetermined coefficients and a Taylor expansion at the fixed point to pin them down *(key insight)*.	Under this parameterization, locate a potential with no counterexample.
4	After the direct proof stalls, propose first proving the zero-temperature case and then lifting it to positive temperature.	Direct proof of the full inequality *(unsuccessful)*; succeed in showing the zero-temperature non-expansion (Theorem 3.3) implies the positive-temperature contraction (Theorem 3.5); attempt at the zero-temperature inequality itself *(unsuccessful)*.
5	Propose expanding the zero-temperature inequality around the center and splitting the domain into a central region and an outer region, certified separately *(key insight)*.	Carry out the exact expansions and produce the Bernstein-basis certificate on each region.
6	*After several iterations: a complete, independently checkable proof.*	

Table 1. The human–AI division of labor across the discovery of the proof. The strategic reframings and the two decisive insights (the Taylor/undetermined-coefficient parameterization in step 3 and the center-versus-boundary domain split in step 5) came from the human side; the model contributed breadth of candidate construction, symbolic computation, and exact certification.

iterations the agent introduced new names, normalizations, and intermediate quantities, often specified only in prose, so that later steps depended on objects whose meaning had shifted—particularly damaging here, where the target is a single global inequality on $\Delta_3 \times \Delta_3$ and a small change of normalization can invalidate the whole claim. In terms of Table 1, the agent could generate the local material of many steps but did not, on its own, supply the strategic reframings of steps 3–5.

4.2. Step 3: adversarial potential search

We next constrained the task to discovering the potential alone, organizing the search—in the spirit of adversarial training—around three roles: a *Potential Constructor* proposing a candidate W for the smallest temperature interval $(w_0, 1]$ it could manage, a *Counterexample Constructor* searching Δ_3^2 (and preferably large w) for violations, and a *Prover* attempting to certify a survivor. The reward structure is natural: the bound tightens as $w \to 0$, so a well-matched pair should converge toward the sharp zero-temperature statement.

In practice the falsifier was the weaker side. Many incorrect potentials passed random and gradient-based numerical checks because their failures sat near structured boundary configurations or in thin directions around u, which the Counterexample Constructor rarely found. The human move that broke the impasse was the step-3 insight: rather than test individual numerical values of the cubic coefficient λ, leave λ undetermined and expand at the fixed point, which forces $\lambda = -3/4$ in one line (Theorem 3.2). The agent was effective at enumerating concrete instances of a fixed form, but the productive move was choosing the parameterization itself—exactly the kind of step Table 1 attributes to the human side.

4.3. Step 5: the final inequality and formal-prover attempts

Once the potential was fixed, the residual task was the finite-dimensional inequality $\mathcal{P} \geq 0$ of Equation (13). We tried to discharge it with two agentic workflows: the Rethlas/Archon loop above (Ju et al., 2026) and a Lean-oriented agent following the Numina-Lean-Agent design (Liu et al., 2026), in both cases reproducing the published workflow rather than invoking a hosted system. Neither found the Bernstein subdivision proof. The Lean-oriented attempts gravitated to general nonlinear-arithmetic tactics such as `nlinarith`, which the expanded inequality defeats through coefficient growth and case explosion; the Rethlas-style attempts again drifted in state without settling on a certificate structure.

What worked was supplying the proof object explicitly (the human step-5 insight): split the domain into a small central region, handled by exact local expansion, and its complement, certified by nonnegative Bernstein coefficients on a rational subdivision. Once that template was fixed, the remaining work—building $\mathcal{P}$, the affine pullbacks, the Bernstein conversion, and the coefficient checks—became reproducible exact arithmetic over $\mathbb{Q}$, and the model was genuinely useful in writing and debugging the verification scripts. This is the shift the case study turns on: from asking an agent to *find a proof* to asking it to *help build and audit a certificate*.

4.4. What the collaboration suggests

Across the three delegated steps the pattern was consistent: in each case the agent supplied useful local material—candidate constructions, algebraic simplifications, verification code—but the move that unblocked the step was a human strategic choice from Table 1. The autonomous search drifted in branch lock-in and definition drift; the adversarial search produced candidates but no certified one, its falsifier missing the thin and boundary violations; the formal attempts fell back on tactics like `nlinarith` that the expanded inequality defeats. The one step that went through—building the exact rational certificate—did so once the central-versus-boundary split was supplied, after which the model was genuinely useful in writing and debugging the scripts.

Read together with Table 1, this points to a consistent and, we think, encouraging picture. Current agents are strong on exactly the parts that are tedious for a human: enumerating candidate forms, carrying out large symbolic expansions, and producing exact rational certificates once the proof object is fixed. They are not yet reliable at the strategic choices—which invariant to seek, which parameterization makes the search tractable, how to decompose the problem—that turned this particular tedious search into a one-line argument. Our workflow is therefore closer to computer-assisted mathematics than to autonomous theorem proving, and the theorem is justified not by any transcript but by the explicit Lyapunov inequality and the independently checkable certificate it reduces to. The practical takeaway is a complementary loop: let the human set the strategy and let the agent supply scale in search and verification, with trust resting on the final exact certificate rather than on the search that produced it.

5. Limitations and Future Work

Limitations. We do not provide a quantitative experimental evaluation of the agentic workflow. We report the qualitative course of the interaction—which moves helped, which failed, and why—but we do not attach success rates, ablation curves, or a standardized benchmark to it. This is partly a deliberate choice: the contribution of an AI system to a research-level proof lies in supplying the right parameterization, the right normalization, and a globally coherent proof architecture, and these are difficult to reduce to a single scalar metric. The mathematical insight that turns a tedious search into a one-line Lyapunov argument is exactly the part that resists quantification, so a leaderboard-style evaluation would risk measuring the easy-to-score surface of the task rather than its substance. We therefore present this as a case study rather than a benchmark, and we regard the design of meaningful quantitative measures for AI-assisted discovery as itself an open problem.

Future work. Two directions seem most worth pursuing, one about how such agents should be built and one about the problem itself.

The first is a better balance between depth and breadth in agentic exploration. What we saw was a recurring pair of failures: an agent would either lock onto one proof branch and keep polishing it long after it had stopped being promising, or skip across many shallow attempts without ever committing to a real investigation. Worse, the longer a run went on, the more it lost track of its own state—names and normalizations drifted, and the agent could no longer tell which subgoals were central and which were dead ends. What seems to be missing is the rhythm a mathematician takes for granted: drilling deep into one calculation while keeping a clear map of what has been settled, what is still open, and when it is time to step back and try something genuinely different. Giving agents that combination—sustained depth, real divergence, and a dependable sense of the global picture—feels like the more important problem than any single tactic.

The second is simply the next case of the problem we studied. Already $q = 4$, $d = 3$ is open, and it is a good deal harder than the $q = 3$, $d = 2$ case treated here. The natural quartic analogue of our potential,

$$V_4(p) = \frac{1}{192}\sum_{i=1}^{4}(4p_i - 3)^4 - \frac{1}{3}, \qquad p \in \Delta_4, \tag{20}$$

is easy to write down, but proving a one-step contraction for it means certifying a polynomial inequality on $\Delta_4 \times \Delta_4 \times \Delta_4$—higher-dimensional and higher-degree than anything we needed here. Whether the search-and-certify workflow of this paper survives that jump is, to us, the most honest test of how far it actually goes.

Impact Statement

This paper presents work whose goal is to understand how AI systems can assist mathematical research. The main risks are epistemic rather than immediate societal harms: overclaiming autonomy, mistaking numerical evidence for proof, and producing unverifiable arguments. We mitigate these risks by separating discovery from certification and by using exact rational verification for the computational component.

References

Alon, N., Bloom, T. F., Gowers, W., Litt, D., Sawin, W., Shankar, A., Tsimerman, J., Wang, V., and Wood, M. M. Remarks on the disproof of the unit distance conjecture. *arXiv preprint arXiv:2605.20695*, 2026.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H.,

Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

de Boer, D., Buys, P., and Regts, G. Uniqueness of the gibbs measure for the 4-state anti-ferromagnetic potts model on the regular tree. *Combinatorics, Probability and Computing*, 32(1):158–182, 2023.

Galanis, A., Goldberg, L. A., and Yang, K. Uniqueness for the 3-state antiferromagnetic potts model on the tree. 2018.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., Santos, E. d. O., et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Google DeepMind. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/, 2025. Accessed: 2026-05-26.

Gu, C., Wu, W., and Yang, K. Power law decay at criticality for the q-state antiferromagnetic Potts model on regular trees. *Electronic Journal of Probability*, 30(none):1 – 25, 2025. doi: 10.1214/25-EJP1277. URL https://doi.org/10.1214/25-EJP1277.

Ju, H., Gao, G., Jiang, J., Wu, B., Sun, Z., Chen, L., Wang, Y., Wang, Y., Wang, Z., He, W., et al. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026.

Liu, J., Zhou, Z., Zhu, Z., Santos, M. D., He, W., Liu, J., Wang, R., Xie, Y., Zhao, J., Wang, Q., et al. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics. *arXiv preprint arXiv:2601.14027*, 2026.

Moura, L. d. and Ullrich, S. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.

Wu, F.-Y. The potts model. *Reviews of modern physics*, 54 (1):235, 1982.

A. Detailed Proofs for Section 3

This appendix contains the technical details suppressed from the proof sketch in Section 3. The computer-assisted parts are exact rational certificates: no floating-point inequality is used in the proof.

A.1. Basic properties of the potential

We prove Theorem 3.1. Write $a_i = p_i - 1/3$. Since $p \in \Delta_3$,

$$-\frac{1}{3} \le a_i \le \frac{2}{3}, \qquad \sum_i a_i = 0,$$

and hence

$$1 - \frac{3}{4}a_i \ge 1 - \frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}.$$

Therefore

$$W(p) = \sum_i a_i^2 \left(1 - \frac{3}{4}a_i\right) \ge \frac{1}{2}\sum_i a_i^2.$$

This proves Equation (8), nonnegativity, and the fact that equality holds only when $a_i = 0$ for all i, i.e. $p = u$.

It remains to check convexity on the affine plane $p_1 + p_2 + p_3 = 1$. Let

$$f(t) = \left(t - \frac{1}{3}\right)^2 - \frac{3}{4}\left(t - \frac{1}{3}\right)^3, \qquad W(p) = \sum_{i=1}^{3} f(p_i).$$

Then

$$f''(t) = \frac{7 - 9t}{2}.$$

For a tangent vector $h \in \mathbb{R}^3$ satisfying $h_1 + h_2 + h_3 = 0$, the second variation is

$$\sum_{i=1}^{3} \frac{7 - 9p_i}{2}\, h_i^2. \tag{21}$$

If all coefficients $(7 - 9p_i)/2$ are nonnegative, the expression is nonnegative. Otherwise exactly one coefficient is negative; by symmetry assume it is the first. Put $b_i = 7 - 9p_i$. Then $b_1 + b_2 + b_3 = 12$, $b_1 < 0$, and $b_2, b_3 > 0$. Setting $h_3 = -h_1 - h_2$, the quadratic form in Equation (21), up to the positive factor $1/2$, has matrix

$$\begin{pmatrix} b_1 + b_3 & b_3 \\ b_3 & b_2 + b_3 \end{pmatrix}.$$

Its first diagonal entry is

$$b_1 + b_3 = 14 - 9(p_1 + p_3) = 5 + 9p_2 > 0.$$

Its determinant is

$$b_1b_2 + b_1b_3 + b_2b_3 = \frac{(b_1 + b_2 + b_3)^2 - (b_1^2 + b_2^2 + b_3^2)}{2}.$$

The function $p \mapsto \sum_i (7 - 9p_i)^2$ is convex, so its maximum over the simplex is attained at a vertex. At a vertex it equals

$$(-2)^2 + 7^2 + 7^2 = 102.$$

Thus

$$b_1b_2 + b_1b_3 + b_2b_3 \ge \frac{144 - 102}{2} > 0.$$

The restricted Hessian is therefore positive semidefinite in every tangent direction, and W is convex on Δ_3.

A.2. Zero-temperature inequality: polynomial reduction

We prove Theorem 3.3. Let

$$G(x,y) = \frac{W(x)+W(y)}{2} - W(\Psi(x,y)), \qquad S(x,y) = \sum_{i=1}^{3}(1-x_i)(1-y_i).$$

For $x, y \in \Delta_3$,

$$S(x,y) = 1 + \sum_{i=1}^{3} x_i y_i \geq 1,$$

so $S > 0$, and $G \geq 0$ is equivalent to

$$\mathcal{P}(x,y) := 8S(x,y)^3 G(x,y) \geq 0.$$

Using

$$4W(p) = 2 - 5\sum_{1\leq i<j\leq 3} p_i p_j - 9p_1p_2p_3,$$

and writing

$$A_i = (1-x_i)(1-y_i), \qquad S = A_1 + A_2 + A_3,$$

one obtains the exact polynomial identity

$$\begin{aligned}\mathcal{P}(x,y) = 2&\left(5S\sum_{1\leq i<j\leq 3} A_iA_j + 9A_1A_2A_3\right)\\ &- S^3\left(5\sum_{1\leq i<j\leq 3} x_ix_j + 9x_1x_2x_3 + 5\sum_{1\leq i<j\leq 3} y_iy_j + 9y_1y_2y_3\right).\end{aligned} \tag{22}$$

Thus the problem is to prove nonnegativity of the polynomial $\mathcal{P}$ on $\Delta_3 \times \Delta_3$. We split the domain into a central box and its complement.

A.3. The central box

Let

$$C_0 = \left\{p \in \Delta_3 : \frac{1}{4} \leq p_i \leq \frac{5}{12} \quad (i = 1,2,3)\right\}.$$

We prove $G \geq 0$ on $C_0 \times C_0$ by an exact local expansion. It is convenient to use real Fourier coordinates. Write

$$p_1 = \frac{1}{3} + \frac{2}{3}\xi, \qquad p_2 = \frac{1}{3} - \frac{1}{3}\xi + \frac{1}{\sqrt{3}}\eta, \qquad p_3 = \frac{1}{3} - \frac{1}{3}\xi - \frac{1}{\sqrt{3}}\eta. \tag{23}$$

Let $\Theta(p) = (\xi, \eta)$. This is the real form of the usual Fourier coordinate $z(p) = p_1 + \omega p_2 + \omega^2 p_3$, with $z = \xi + i\eta$. A direct substitution gives

$$W(p) = \frac{2}{3}(\xi^2 + \eta^2) - \frac{1}{6}(\xi^3 - 3\xi\eta^2). \tag{24}$$

If $p_i = 1/3 + h_i$, then

$$\xi^2 + \eta^2 = \frac{3}{2}\sum_{i=1}^{3} h_i^2.$$

For $p \in C_0$, $|h_i| \leq 1/12$ and $\sum_i h_i = 0$, hence

$$\xi^2 + \eta^2 \leq \frac{1}{48} < \frac{1}{36}. \tag{25}$$

Thus $\|\Theta(p)\| < 1/6$ on C_0.

For $x, y \in C_0$, define

$$\Theta(x) = m + d, \qquad \Theta(y) = m - d,$$

where $m = (m_1, m_2)$ and $d = (d_1, d_2)$. From Equation (25),

$$\|m\| \le \frac{1}{6}, \qquad \|d\| \le \frac{1}{6}. \tag{26}$$

Set

$$R = 4 + 2(\|m\|^2 - \|d\|^2).$$

Then $R \ge 4 - 2/36 = 71/18 > 0$. Exact substitution of Equation (23) into the zero-temperature recursion gives

$$\Theta(\Psi(x, y)) = (Q_1, Q_2), \tag{27}$$

where

$$Q_1 = \frac{m_1^2 - m_2^2 - d_1^2 + d_2^2 - 4m_1}{R}, \tag{28}$$

$$Q_2 = \frac{-2m_1m_2 + 2d_1d_2 - 4m_2}{R}. \tag{29}$$

Equivalently, in complex notation this is

$$z(\Psi(x, y)) = \frac{\overline{m}^2 - \overline{d}^2 - 4m}{4 + 2(|m|^2 - |d|^2)}. \tag{30}$$

Both forms are exact algebraic identities.

Define

$$K(m_1, m_2, d_1, d_2) = 6R^3 G(x, y).$$

Since $R > 0$, proving $K \ge 0$ is equivalent to proving $G \ge 0$. Substituting Equations (24) and (27) and expanding over $\mathbb{Q}$, we obtain a polynomial identity

$$K = K_0(m) + K_2(m, d) + K_{\ge 4}(m, d), \tag{31}$$

where the decomposition is by total degree in (d_1, d_2). There are no odd d-degree terms, because exchanging x and y fixes m, sends d to $-d$, and leaves G unchanged.

Let $r = \|m\|$. The d-independent part is

$$\begin{aligned} K_0(m) = {} & 288r^4 - 44m_1(m_1^2 - 3m_2^2)r^2 + K_6(m) \\ & - 48m_1(m_1^2 - 3m_2^2)r^4 + 32r^8 - 8m_1(m_1^2 - 3m_2^2)r^6, \end{aligned} \tag{32}$$

where

$$K_6(m) = 185m_1^6 + 537m_1^4m_2^2 + 567m_1^2m_2^4 + 183m_2^6 \ge 0.$$

Writing $m_1 = r\cos\theta$, $m_2 = r\sin\theta$ gives

$$|m_1(m_1^2 - 3m_2^2)| = r^3|\cos 3\theta| \le r^3.$$

Since $r \le 1/6$,

$$K_0(m) \ge r^4 \left(288 - \frac{44}{6} - \frac{48}{6^3} - \frac{8}{6^5}\right) \ge 0. \tag{33}$$

The final inequality is non-strict to include $r = 0$.

The quadratic part in d has the form

$$K_2(m, d) = A(m)d_1^2 + B(m)d_1d_2 + C(m)d_2^2. \tag{34}$$

The explicit polynomials are

$$\begin{aligned}
A(m) - 256 &= -64m_1^6 - 96m_1^5m_2^2 - 48m_1^5 - 192m_1^4m_2^2 - 171m_1^4 \\
&\quad - 192m_1^3m_2^4 - 480m_1^3m_2^2 - 296m_1^3 - 192m_1^2m_2^4 \\
&\quad - 350m_1^2m_2^2 + 112m_1^2 - 96m_1m_2^6 - 432m_1m_2^4 \\
&\quad - 424m_1m_2^2 - 320m_1 - 64m_2^6 - 203m_2^4 + 144m_2^2, \\
B(m) &= 48m_1^6m_2 + 144m_1^4m_2^3 + 288m_1^4m_2 + 88m_1^3m_2 \\
&\quad + 144m_1^2m_2^5 + 576m_1^2m_2^3 + 656m_1^2m_2 + 40m_1m_2^3 \\
&\quad - 64m_1m_2 + 48m_2^7 + 288m_2^5 + 656m_2^3 + 640m_2, \\
C(m) - 256 &= 48m_1^7 - 64m_1^6 + 48m_1^5m_2^2 + 240m_1^5 - 192m_1^4m_2^2 \\
&\quad - 197m_1^4 - 48m_1^3m_2^4 + 96m_1^3m_2^2 + 360m_1^3 \\
&\quad - 192m_1^2m_2^4 - 386m_1^2m_2^2 + 144m_1^2 \\
&\quad - 48m_1m_2^6 - 144m_1m_2^4 + 232m_1m_2^2 + 320m_1 \\
&\quad - 64m_2^6 - 165m_2^4 + 112m_2^2.
\end{aligned}$$

For a polynomial $Q(m_1, m_2) = \sum q_{ab} m_1^a m_2^b$, define

$$\|Q\|_{1,1/6} = \sum_{a,b} |q_{ab}| 6^{-(a+b)}.$$

On the box $|m_1|, |m_2| \le 1/6$, the triangle inequality gives $|Q(m)| \le \|Q\|_{1,1/6}$. Exact coefficient summation gives

$$\|A - 256\|_{1,1/6} = \frac{2321}{36}, \qquad \|B\|_{1,1/6} = \frac{83665}{729}, \qquad \|C - 256\|_{1,1/6} = \frac{186145}{2916}.$$

Using $|d_1 d_2| \le (d_1^2 + d_2^2)/2$, we obtain

$$K_2(m, d) \ge \frac{391165}{2916}(d_1^2 + d_2^2) = \frac{391165}{2916}\|d\|^2. \tag{35}$$

It remains to control the higher d-degree terms. Write

$$K_{\ge 4} = \sum q_{abce} m_1^a m_2^b d_1^c d_2^e, \qquad c + e \ge 4.$$

Using $|m_1|, |m_2| \le 1/6$, $|d_1|, |d_2| \le \|d\|$, and $\|d\| \le 1/6$, each monomial obeys

$$|m_1^a m_2^b d_1^c d_2^e| \le 6^{-(a+b+c+e-2)} \|d\|^2.$$

The exact coefficient sum is

$$\sum |q_{abce}| 6^{-(a+b+c+e-2)} = \frac{337601}{5832}.$$

Hence

$$|K_{\ge 4}(m, d)| \le \frac{337601}{5832}\|d\|^2. \tag{36}$$

Combining Equations (33), (35) and (36) gives

$$K \ge r^4 \left(288 - \frac{44}{6} - \frac{48}{6^3} - \frac{8}{6^5}\right) + \left(\frac{391165}{2916} - \frac{337601}{5832}\right)\|d\|^2 \ge 0.$$

Since

$$288 - \frac{44}{6} - \frac{48}{6^3} - \frac{8}{6^5} > 0, \qquad \frac{391165}{2916} - \frac{337601}{5832} = \frac{148243}{1944} > 0,$$

this gives $G \ge 0$ on $C_0 \times C_0$. These two rational inequalities, together with the polynomial identities for $K_0, K_2, K_{\ge 4}$ and the coefficient-sum bounds, are verified in exact arithmetic by the script of Section B.1.

A.4. Bernstein subdivision on the complement

We now prove $\mathcal{P} \geq 0$ on $(\Delta_3 \times \Delta_3) \setminus (C_0 \times C_0)$. Put

$$x = (x_1, x_2, 1 - x_1 - x_2), \qquad y = (y_1, y_2, 1 - y_1 - y_2).$$

The certificate represents the numerator of $8S^3G$ after this substitution, namely $\widetilde{\mathcal{P}} = 9\mathcal{P}$. Since $9 > 0$, this has the same sign as $\mathcal{P}$.

We use the following standard fact. If a polynomial on a triangle T, after affine pullback to the reference triangle, has all coefficients nonnegative in the degree-six Bernstein basis, then the polynomial is nonnegative on T. This is because Bernstein basis functions are nonnegative on the triangle. On product cells $T_x \times T_y$, we use the tensor-product Bernstein basis. Multiplying all coefficients by positive common denominators preserves their signs.

The monomial-to-Bernstein conversion on the reference triangle is

$$r^a s^b = \sum_{i+j \leq 6} \frac{\binom{i}{a}\binom{j}{b}}{\binom{6}{a+b}\binom{a+b}{a}} B^6_{(i,j,6-i-j)}(r,s), \tag{37}$$

where the summand is zero unless $a \leq i$ and $b \leq j$. Applying Equation (37) in both factors converts every product-cell pullback of $\widetilde{\mathcal{P}}$ into degree-six tensor-product Bernstein form.

The triangulation is as follows. First triangulate Δ_3 by the rational grid of mesh $1/6$, giving 36 triangles. Let

$$C_1 = \left\{ p \in \Delta_3 : \frac{1}{6} \leq p_i \leq \frac{1}{2} \quad (i = 1, 2, 3) \right\}.$$

Exactly six of the 36 triangles are contained in C_1. On every product cell for which at least one factor triangle is not contained in C_1, the degree-six tensor-product Bernstein expansion of $\widetilde{\mathcal{P}}$ has all coefficients nonnegative. This checks

$$36^2 - 6^2 = 1260$$

product cells.

The six triangles contained in C_1 are then split once by joining edge midpoints. This gives 24 subtriangles, six of which are contained in C_0. Product cells lying entirely in $C_0 \times C_0$ are handled by Section A.3; all remaining annular product cells are checked by the same Bernstein test. There are

$$24^2 - 6^2 = 540$$

such product cells. Eighteen of them have a negative Bernstein coefficient at this intermediate resolution; each of these product cells is split once more in both factors, producing

$$18 \cdot 4 \cdot 4 = 288$$

final product cells. On all of these final cells, all degree-six tensor-product Bernstein coefficients are nonnegative.

The verifier constructs $\widetilde{\mathcal{P}}$ directly from the definitions of W and Ψ (equivalently, from Equation (22)), enumerates the rational subdivision cells, applies the affine pullback on each product cell, converts to degree-six tensor-product Bernstein form via Equation (37), and checks nonnegativity of every coefficient in exact rational arithmetic. No coefficient is read from floating-point output. The certificate reports

```
polynomial terms: 420
Bernstein degree per simplex: 6
level-1 triangles: 36
large central triangles: 6
small analytic triangles after split: 6
outside-large-box product cells checked: 1260
outside-large-box negative Bernstein coefficients: 0
annulus product cells checked before final split: 540
```

```
annulus bad product cells requiring final split: 18
final split product cells checked: 288
final split negative Bernstein coefficients: 0
certificate status: PASS
```

All arithmetic is over $\mathbb{Q}$. Hence $\widetilde{\mathcal{P}} \geq 0$, and therefore $\mathcal{P} \geq 0$, on the complement of $C_0 \times C_0$. Together with the central-box proof, $\mathcal{P} \geq 0$ everywhere. Since $\mathcal{P} = 8S^3G$ and $S > 0$, $G \geq 0$ on $\Delta_3 \times \Delta_3$, proving Theorem 3.3. The full script is listed in Section B.2.

A.5. Positive temperature and the tree iteration

We first prove Theorem 3.4. For each coordinate i,

$$\alpha L_0(x)_i + \beta u_i = \frac{2(1-w)}{2+w} \cdot \frac{1-x_i}{2} + \frac{3w}{2+w} \cdot \frac{1}{3} = \frac{1-(1-w)x_i}{2+w} = L_w(x)_i.$$

This proves the decomposition.

We next prove Theorem 3.5. Let $a = L_0(x)$, $b = L_0(y)$. Then

$$L_w(x) = \alpha a + \beta u, \qquad L_w(y) = \alpha b + \beta u.$$

Expanding the pointwise product gives

$$L_w(x)_i L_w(y)_i = \alpha^2 a_i b_i + \alpha\beta a_i u_i + \alpha\beta u_i b_i + \beta^2 u_i^2.$$

After normalization, $\Phi_w(x, y)$ is a convex combination

$$\Phi_w(x, y) = \lambda_{11} p_{11} + \lambda_{10} p_{10} + \lambda_{01} p_{01} + \lambda_{00} p_{00},$$

where

$$p_{11} = \text{Normalize}(a \odot b) = \Psi(x, y), \quad p_{10} = \text{Normalize}(a \odot u) = a = \Psi(x, u),$$
$$p_{01} = \text{Normalize}(u \odot b) = b = \Psi(u, y), \quad p_{00} = \text{Normalize}(u \odot u) = u.$$

Let

$$Z = \sum_{i=1}^{3} L_w(x)_i L_w(y)_i.$$

Since $L_w(x)$ and $L_w(y)$ are probability vectors, $Z \leq 1$. The relevant weights are

$$\lambda_{10} = \frac{\alpha\beta/3}{Z}, \qquad \lambda_{01} = \frac{\alpha\beta/3}{Z}, \qquad \lambda_{00} = \frac{\beta^2/3}{Z}.$$

By convexity of W, Theorem 3.3 applied with u (which gives $W(p_{10}) = W(\Psi(x, u)) \leq \frac{1}{2}(W(x) + W(u)) = \frac{1}{2}W(x)$ and likewise $W(p_{01}) \leq \frac{1}{2}W(y)$), and $W(u) = 0$, if $M = \max\{W(x), W(y)\}$ then

$$\begin{aligned} W(\Phi_w(x, y)) &\leq \lambda_{11} W(p_{11}) + \lambda_{10} W(p_{10}) + \lambda_{01} W(p_{01}) + \lambda_{00} W(p_{00}) \\ &\leq \left(\lambda_{11} + \frac{\lambda_{10} + \lambda_{01}}{2}\right) M \\ &= \left(1 - \lambda_{00} - \frac{\lambda_{10} + \lambda_{01}}{2}\right) M. \end{aligned}$$

Moreover,

$$\lambda_{00} + \frac{\lambda_{10} + \lambda_{01}}{2} = \frac{\beta^2/3 + \alpha\beta/3}{Z} = \frac{\beta(\alpha + \beta)}{3Z} = \frac{\beta}{3Z} \geq \frac{\beta}{3}.$$

Therefore

$$W(\Phi_w(x, y)) \leq \left(1 - \frac{\beta}{3}\right) M = \frac{2}{2+w} M,$$

which proves Theorem 3.5.

Finally, we prove Theorem 3.6. Let M_h be the largest possible value of W at the root of a height-h binary subtree over all boundary conditions. At height 0, the message is a delta mass, and $W(\delta_i) = 1/2$, so $M_0 \le 1/2$. By Theorem 3.5,

$$M_{h+1} \le \frac{2}{2+w} M_h, \qquad M_n \le \frac{1}{2}\left(\frac{2}{2+w}\right)^n.$$

For a boundary condition τ, the root message is exactly the root marginal:

$$m^{\tau}_{r_n}(i) = \mu^{\tau}_{n,w}(\sigma(r_n) = i).$$

By Equation (8), for each color c,

$$\begin{aligned}\left|\mu^{\tau}_{n,w}(\sigma(r_n) = c) - \frac{1}{3}\right| &\le \left(\sum_{i=1}^{3}\left(m^{\tau}_{r_n}(i) - \frac{1}{3}\right)^2\right)^{1/2} \\ &\le \left(2W(m^{\tau}_{r_n})\right)^{1/2} \\ &\le \left(\frac{2}{2+w}\right)^{n/2}.\end{aligned}$$

Taking the maximum over τ proves Equation (19); since $2/(2+w) < 1$, the boundary influence decays to zero exponentially.

B. Verification Code Included in the Certificate

This appendix prints the two exact verification scripts used in Theorem 3.3. The scripts are part of the certificate rather than numerical evidence. They reconstruct the polynomials from the definitions, use exact rational arithmetic, and terminate with assertions that match the inequalities used in Sections A.3 and A.4. The only use of `numpy` in the Bernstein script is matrix multiplication with `dtype=object`; the entries are Python integers obtained from rational coefficients after multiplication by positive common denominators, so no floating-point linear algebra is involved.

B.1. Central-box algebra certificate

The first script verifies the real polynomial identity for $K = 6R^3G$, the displayed decomposition $K = K_0 + K_2 + K_{\ge 4}$, and the exact coefficient-sum bounds in Equations (33), (35) and (36).

```python
"""Exact algebra check for the central-box Fourier/Taylor part.

This script reconstructs the polynomial
   K = 6 R^3 ((W(x)+W(y))/2 - W(Psi(x,y)))
in the Fourier coordinates used in the proof, splits it according to degree in
(d1,d2), and verifies the rational coefficient-sum bounds appearing in the
central-box estimate.

All arithmetic is exact SymPy rational arithmetic. No floating point numbers are
used in the certification.
"""

import sympy as sp

m1, m2, d1, d2 = sp.symbols("m1 m2 d1 d2", real=True)
I = sp.I
m = m1 + I * m2
d = d1 + I * d2
mc = m1 - I * m2
dc = d1 - I * d2
R = 4 + 2 * ((m1**2 + m2**2) - (d1**2 + d2**2))


def W_ab(a, b):
   """W in the Fourier coordinate w=a+ib: W=(2/3)|w|^2-(1/6)Re(w^3)."""
   return sp.Rational(2, 3) * (a * a + b * b) - sp.Rational(1, 6) * (a**3 - 3 * a * b**2)


# z(x)=m+d and z(y)=m-d.
```

```python
W_x = W_ab(m1 + d1, m2 + d2)
W_y = W_ab(m1 - d1, m2 - d2)

# z(Psi)=N/R where N=conj(m)^2-conj(d)^2-4m.
N = sp.expand(mc**2 - dc**2 - 4 * m)
Nr = sp.expand(sp.re(N))
Ni = sp.expand(sp.im(N))
W_psi = sp.Rational(2, 3) * (Nr**2 + Ni**2) / R**2 - sp.Rational(1, 6) * (Nr**3 - 3 * Nr * Ni**2) / R**3

G = sp.Rational(1, 2) * (W_x + W_y) - W_psi
K_expr = sp.cancel(6 * R**3 * G)
K_num, K_den = sp.fraction(K_expr)
assert sp.factor(K_den) == 1
K = sp.Poly(sp.expand(K_num), m1, m2, d1, d2)

K0 = 0
K2 = 0
K_ge4 = 0
A = 0
B = 0
C = 0

for (a, b, c, e), coeff in K.terms():
   d_degree = c + e
   monomial = coeff * m1**a * m2**b * d1**c * d2**e
   if d_degree == 0:
      K0 += monomial
   elif d_degree == 2:
      K2 += monomial
      if c == 2:
         A += coeff * m1**a * m2**b
      elif c == 1 and e == 1:
         B += coeff * m1**a * m2**b
      elif e == 2:
         C += coeff * m1**a * m2**b
      else:
         raise AssertionError("unexpected d-degree-two monomial")
   elif d_degree >= 4:
      K_ge4 += monomial
   else:
      raise AssertionError("unexpected odd d-degree")

# Check K0 against the displayed grouped expression.
r2 = m1**2 + m2**2
re_m3 = m1 * (m1**2 - 3 * m2**2)
K6 = 185 * m1**6 + 537 * m1**4 * m2**2 + 567 * m1**2 * m2**4 + 183 * m2**6
K0_display = 288 * r2**2 - 44 * re_m3 * r2 + K6 - 48 * re_m3 * r2**2 + 32 * r2**4 - 8 * re_m3 * r2**3
assert sp.expand(K0 - K0_display) == 0

A_expected = (
   -64*m1**6 - 96*m1**5*m2**2 - 48*m1**5 - 192*m1**4*m2**2 - 171*m1**4
   - 192*m1**3*m2**4 - 480*m1**3*m2**2 - 296*m1**3 - 192*m1**2*m2**4
   - 350*m1**2*m2**2 + 112*m1**2 - 96*m1*m2**6 - 432*m1*m2**4
   - 424*m1*m2**2 - 320*m1 - 64*m2**6 - 203*m2**4 + 144*m2**2 + 256
)
B_expected = (
   48*m1**6*m2 + 144*m1**4*m2**3 + 288*m1**4*m2 + 88*m1**3*m2
   + 144*m1**2*m2**5 + 576*m1**2*m2**3 + 656*m1**2*m2 + 40*m1*m2**3
   - 64*m1*m2 + 48*m2**7 + 288*m2**5 + 656*m2**3 + 640*m2
)
C_expected = (
   48*m1**7 - 64*m1**6 + 48*m1**5*m2**2 + 240*m1**5 - 192*m1**4*m2**2
   - 197*m1**4 - 48*m1**3*m2**4 + 96*m1**3*m2**2 + 360*m1**3
   - 192*m1**2*m2**4 - 386*m1**2*m2**2 + 144*m1**2 - 48*m1*m2**6
   - 144*m1*m2**4 + 232*m1*m2**2 + 320*m1 - 64*m2**6 - 165*m2**4
   + 112*m2**2 + 256
)
assert sp.expand(A - A_expected) == 0
assert sp.expand(B - B_expected) == 0
assert sp.expand(C - C_expected) == 0

rho = sp.Rational(1, 6)


def l1_radius(poly, variables, radius):
   P = sp.Poly(sp.expand(poly), *variables)
   total = sp.Rational(0)
   for exps, coeff in P.terms():
      total += abs(coeff) * radius ** sum(exps)
   return sp.factor(total)

```

```python

def d_ge4_bound(poly):
    P = sp.Poly(sp.expand(poly), m1, m2, d1, d2)
    total = sp.Rational(0)
    for (a, b, c, e), coeff in P.terms():
        d_degree = c + e
        assert d_degree >= 4
        total += abs(coeff) * rho ** (a + b + d_degree - 2)
    return sp.factor(total)

A_bound = l1_radius(A - 256, (m1, m2), rho)
B_bound = l1_radius(B, (m1, m2), rho)
C_bound = l1_radius(C - 256, (m1, m2), rho)
K_ge4_bound = d_ge4_bound(K_ge4)

assert A_bound == sp.Rational(2321, 36)
assert B_bound == sp.Rational(83665, 729)
assert C_bound == sp.Rational(186145, 2916)
assert K_ge4_bound == sp.Rational(337601, 5832)

print("central polynomial terms:", len(K.terms()))
print("A bound:", A_bound)
print("B bound:", B_bound)
print("C bound:", C_bound)
print("K_ge4 bound:", K_ge4_bound)
print("central-box algebra certificate: PASS")
```

Listing 1. Exact central-box algebra certificate.

B.2. Bernstein subdivision certificate

The second script verifies $\widetilde{\mathcal{P}} = 9\mathcal{P}$ on the complement of $C_0 \times C_0$. It implements the degree-six monomial-to-Bernstein conversion Equation (37), performs the rational triangulation and the finite subdivision described in Section A.4, and asserts that every final checked cell has no negative Bernstein coefficient.

```python
"""Exact Bernstein certificate for the q=3 binary-tree AF Potts zero-temperature inequality.

It certifies the non-analytic part of
    W(Psi(x,y)) <= (W(x)+W(y))/2, x,y in Delta_3.

The small central box
    C0 = {p in Delta_3 : 1/4 <= p_i <= 5/12 for i=1,2,3}
is handled by the analytic Fourier/Taylor lemma in the replacement proof.
This script proves the inequality on the complement of C0 x C0 using exact
rational Bernstein coefficients.

All arithmetic used for certification is exact. Floating point is not used.
"""
from fractions import Fraction
from math import comb, lcm
import time
from functools import lru_cache
import numpy as np
import sympy as sp

# ---------------------------- polynomial ---------------------------------
x1, x2, y1, y2 = sp.symbols("x1 x2 y1 y2")
x = [x1, x2, 1 - x1 - x2]
y = [y1, y2, 1 - y1 - y2]
S = sum((1 - x[i]) * (1 - y[i]) for i in range(3))
z = [(1 - x[i]) * (1 - y[i]) / S for i in range(3)]

def W(p):
    return (
        sum((pi - sp.Rational(1, 3)) ** 2 for pi in p)
        - sp.Rational(3, 4) * sum((pi - sp.Rational(1, 3)) ** 3 for pi in p)
    )

G = sp.Rational(1, 2) * W(x) + sp.Rational(1, 2) * W(y) - W(z)
P_expr = sp.together(8 * S**3 * G).as_numer_denom()[0]
P = sp.Poly(sp.expand(P_expr), x1, x2, y1, y2)

def sp_to_fraction(q):
    q = sp.Rational(q)
    return Fraction(int(q.p), int(q.q))

```

```python
terms = [((a, b, c, d), sp_to_fraction(coeff)) for (a, b, c, d), coeff in P.terms()]

# ----------------------- Bernstein conversion ----------------------------
n = 6
basis = [(i, j) for i in range(n + 1) for j in range(n + 1 - i)]
nb = len(basis)
bern_factors = {}
for a in range(n + 1):
    for b in range(n + 1 - a):
        den = comb(n, a + b) * comb(a + b, a) # crucial multinomial factor
        vals = []
        for idx, (i, j) in enumerate(basis):
            if a <= i and b <= j:
                vals.append((idx, Fraction(comb(i, a) * comb(j, b), den)))
        bern_factors[(a, b)] = vals

x_exps = sorted({(a, b) for (a, b, c, d), _ in terms})
y_exps = sorted({(c, d) for (a, b, c, d), _ in terms})
x_index = {e: i for i, e in enumerate(x_exps)}
y_index = {e: i for i, e in enumerate(y_exps)}

C = [[Fraction(0) for _ in y_exps] for __ in x_exps]
for (a, b, c, d), coeff in terms:
    C[x_index[(a, b)]][y_index[(c, d)]] += coeff
assert all(v.denominator == 1 for row in C for v in row)
C_int = np.array([[int(v) for v in row] for row in C], dtype=object)

# -------------------------- exact utilities -------------------------------
def poly_mul(p, q):
    out = {}
    for (a, b), ca in p.items():
        for (c, d), cb in q.items():
            out[(a + c, b + d)] = out.get((a + c, b + d), Fraction(0)) + ca * cb
    return {k: v for k, v in out.items() if v}

def poly_pow_linear(a0, a1, a2, power):
    out = {(0, 0): Fraction(1)}
    base = {}
    if a0:
        base[(0, 0)] = a0
    if a1:
        base[(1, 0)] = a1
    if a2:
        base[(0, 1)] = a2
    for _ in range(power):
        out = poly_mul(out, base)
    return out

def power_to_bernstein(poly):
    vec = [Fraction(0) for _ in basis]
    for (a, b), coeff in poly.items():
        for idx, fac in bern_factors[(a, b)]:
            vec[idx] += coeff * fac
    return vec

def monomial_pullback_bernstein(tri, exp):
    v0, v1, v2 = tri
    a0 = v0[0]
    a1 = v1[0] - v0[0]
    a2 = v2[0] - v0[0]
    b0 = v0[1]
    b1 = v1[1] - v0[1]
    b2 = v2[1] - v0[1]
    poly = poly_mul(poly_pow_linear(a0, a1, a2, exp[0]),
                    poly_pow_linear(b0, b1, b2, exp[1]))
    return power_to_bernstein(poly)

@lru_cache(maxsize=None)
def B_int_matrix_for_triangle_cached(tri, exps_tuple):
    exps = list(exps_tuple)
    frac_mat = [[Fraction(0) for _ in exps] for __ in range(nb)]
    den = 1
    for col, exp in enumerate(exps):
        vec = monomial_pullback_bernstein(tri, exp)
        for row, val in enumerate(vec):
            frac_mat[row][col] = val
            den = lcm(den, val.denominator)
    arr = np.empty((nb, len(exps)), dtype=object)
    for i in range(nb):
        for j in range(len(exps)):
            val = frac_mat[i][j]
```

```
            arr[i, j] = val.numerator * (den // val.denominator)
    return arr

def B_int_matrix_for_triangle(tri, exps):
    return B_int_matrix_for_triangle_cached(tri, tuple(exps))

def check_product_cell(tri_x, tri_y):
    Bx = B_int_matrix_for_triangle(tri_x, x_exps)
    By = B_int_matrix_for_triangle(tri_y, y_exps)
    coeffs = (Bx @ C_int) @ By.T
    flat = list(coeffs.ravel())
    mn = min(flat)
    nneg = sum(1 for v in flat if v < 0)
    return nneg, mn

# ------------------------- triangulations --------------------------------
def simplex_tris(N):
    tris = []
    for i in range(N):
        for j in range(N - i):
            v0 = (Fraction(i, N), Fraction(j, N))
            v1 = (Fraction(i + 1, N), Fraction(j, N))
            v2 = (Fraction(i, N), Fraction(j + 1, N))
            tris.append((v0, v1, v2))
            if i + j <= N - 2:
                v0 = (Fraction(i + 1, N), Fraction(j, N))
                v1 = (Fraction(i + 1, N), Fraction(j + 1, N))
                v2 = (Fraction(i, N), Fraction(j + 1, N))
                tris.append((v0, v1, v2))
    return tris

def tri_inside_box(tri, lo, hi):
    for v in tri:
        p = [v[0], v[1], Fraction(1) - v[0] - v[1]]
        if not all(lo <= q <= hi for q in p):
            return False
    return True

def split_tri4(tri):
    v0, v1, v2 = tri
    def mid(a, b):
        return ((a[0] + b[0]) / 2, (a[1] + b[1]) / 2)
    m01 = mid(v0, v1)
    m12 = mid(v1, v2)
    m20 = mid(v2, v0)
    return [(v0, m01, m20), (m01, v1, m12), (m20, m12, v2), (m01, m12, m20)]

# ---------------------------- certificate --------------------------------
def main():
    t0 = time.time()
    big_lo, big_hi = Fraction(1, 6), Fraction(1, 2)
    small_lo, small_hi = Fraction(1, 4), Fraction(5, 12)

    # Level 1: N=6 triangulation. Certify all cells outside the larger central box.
    tris6 = simplex_tris(6)
    big = [tri_inside_box(T, big_lo, big_hi) for T in tris6]
    B6 = [B_int_matrix_for_triangle(T, x_exps) for T in tris6]
    L6 = [B @ C_int for B in B6]

    outside_checked = 0
    outside_negative = 0
    outside_min = None
    for ix, L in enumerate(L6):
        for iy, By in enumerate(B6):
            if big[ix] and big[iy]:
                continue
            coeffs = L @ By.T
            flat = list(coeffs.ravel())
            mn = min(flat)
            outside_min = mn if outside_min is None else min(outside_min, mn)
            nneg = sum(1 for v in flat if v < 0)
            outside_negative += nneg
            outside_checked += 1
            assert nneg == 0, ("outside big box failed", ix, iy, nneg, mn)

    # Level 2: inside the larger central box, split its six triangles once.
    big_tris = [T for T, flag in zip(tris6, big) if flag]
    subtris = []
    for T in big_tris:
        subtris.extend(split_tri4(T))
    small = [tri_inside_box(T, small_lo, small_hi) for T in subtris]
```

```python

    annulus_checked = 0
    annulus_negative = 0
    annulus_min = None
    bad_annulus = []
    for ix, tx in enumerate(subtris):
        for iy, ty in enumerate(subtris):
            if small[ix] and small[iy]:
                continue # analytic central box
            nneg, mn = check_product_cell(tx, ty)
            annulus_checked += 1
            annulus_min = mn if annulus_min is None else min(annulus_min, mn)
            if nneg == 0:
                continue
            annulus_negative += nneg
            bad_annulus.append((ix, iy, tx, ty, nneg, mn))

    # Level 3: split the finitely many bad annulus cells once more.
    split_checked = 0
    split_negative = 0
    split_min = None
    for ix, iy, tx, ty, _, _ in bad_annulus:
        for sx in split_tri4(tx):
            for sy in split_tri4(ty):
                if tri_inside_box(sx, small_lo, small_hi) and tri_inside_box(sy, small_lo, small_hi):
                    continue
                nneg, mn = check_product_cell(sx, sy)
                split_checked += 1
                split_min = mn if split_min is None else min(split_min, mn)
                split_negative += nneg
                assert nneg == 0, ("split annulus failed", ix, iy, sx, sy, nneg, mn)

    print("polynomial terms:", len(terms))
    print("Bernstein degree per simplex:", n)
    print("level-1 triangles:", len(tris6))
    print("large central triangles:", sum(big))
    print("small analytic triangles after split:", sum(small))
    print("outside-large-box product cells checked:", outside_checked)
    print("outside-large-box negative Bernstein coefficients:", outside_negative)
    print("outside-large-box minimum scaled coefficient:", outside_min)
    print("annulus product cells checked before final split:", annulus_checked)
    print("annulus bad product cells requiring final split:", len(bad_annulus))
    print("annulus negative Bernstein coefficients before final split:", annulus_negative)
    print("annulus minimum scaled coefficient before final split:", annulus_min)
    print("final split product cells checked:", split_checked)
    print("final split negative Bernstein coefficients:", split_negative)
    print("final split minimum scaled coefficient:", split_min)
    print("certificate status: PASS")
    print("elapsed seconds: %.2f" % (time.time() - t0))

if __name__ == "__main__":
    main()
```

Listing 2. Exact Bernstein subdivision certificate.

C. Discovery Logs and Experimental Details

The exact rational verification scripts of Section B are provided as supplementary material: `verify_central.py` (the central-box algebra of Section A.3) and `verify_bernstein.py` (the subdivision certificate of Section A.4). Both use exact rational arithmetic; no inequality is decided in floating point. Running `python verify_central.py && python verify_bernstein.py` reproduces every coefficient bound and the certificate counts reported above. The SHA256 hash of the archive is `<insert-hash>`. The supplementary archive also contains representative transcripts of the adversarial search loop—candidate potential generation, numerical counterexamples, the transition to an exact certificate, and failed proof attempts. The proof above does not rely on the transcripts; they serve only to document the discovery process.

D. Additional Experimental Details

Implementation details to report in an experimental version include optimization seeds, boundary-biased sampling schemes, symbolic simplification steps, exact arithmetic libraries, certificate runtime, and hardware. Negative controls should include the pure quadratic potential and the one-parameter cubic family before the Taylor-forced choice $\lambda = -3/4$.