QED: An Open-Source Multi-Agent System for Generating Mathematical Proofs on Open Problems

Chenyang An [1] **Qihao Ye** [1] **Minghao Pan** [2] **Jiayun Zhang** [3]

Abstract

We present **QED**, an open-source multi-agent system that turns human-provided research questions into complete mathematical proofs without further human guidance. Its pipeline is designed to overcome common failures of single-query proof generation by separating planning, proving, and verification: a decomposition agent structures the proof search, prover agents generate candidate arguments, and verifier agents check correctness. In collaboration with domain experts, we evaluated QED on 18 research-level projects of varying difficulty. QED produced five original works across algebraic geometry, fluid PDEs, probability, and inverse problems. Expert assessments regard these works as solid specialized research contributions, with three comparable in difficulty and scope to work commonly published in established specialist mathematics venues. QED is released at `https://github.com/proofQED/QED`.

1. Introduction

Large language models (LLMs) have demonstrated strong and rapidly improving performance on mathematical reasoning, progressing from grade-school word problems (Cobbe et al., 2021) and competition mathematics (Hendrycks et al., 2021; Trinh et al., 2024; Guo et al., 2025) toward research-level use. Recent reports include individual mathematicians using LLM chatbots to obtain core ideas for ongoing research (Gowers, 2026; Alexeev et al., 2026a), as well as large-scale systems such as Aletheia (Feng et al., 2026), which have contributed to mathematical discovery with varying degrees of human guidance. Fully automatic proof generation from only a human-provided problem statement remains rarer. Notable examples include Aletheia's autonomous results, including (Feng, 2026), OpenAI's short solutions to Erdős problems (Alexeev et al., 2026b;c) and OpenAI's recent announcement that an internal reasoning model produced a counterexample to a longstanding conjecture in the planar unit-distance problem (OpenAI, 2026). Google DeepMind's AI Co-Mathematician further explores human-in-the-loop mathematical research through a Gemini-based, stateful multi-agent workbench (Zheng et al., 2026). These developments mark a shift from benchmark problem solving toward AI-assisted mathematical research. However, most high-profile demonstrations still rely on closed systems, models that are not publicly released, or human-directed workflows. This motivates **QED**, an open-source, fully automatic multi-agent proof system evaluated on expert-proposed research problems.

Through experiments with frontier coding agents (Claude Code, Codex, and Gemini CLI) on research-level proof tasks, we observe that naive deployment (single-turn prompting, unchecked outputs, no structured verification) leads to identifiable failure modes (Section 3). We characterize seven such failure modes and demonstrate that each can be mitigated through a targeted architectural intervention. Each architectural design choice in QED is directly motivated by an observed failure mode:

1. **Context contamination**: Prover and verifier share context, producing circular reasoning → *separated prover and verifier agents*.
2. **Citation hallucination**: Models fabricate or misstate referenced theorems → *structured citation verification with exact-statement matching*.
3. **Misallocation of proof effort**: Models skip the hardest parts of proofs while continuing to work on easy steps → *mandatory* `<key-original-step>` *tagging with verifier enforcement*.
4. **Unstable proof plans**: Models abandon strategies mid-proof on hard problems → *decomposition mode separating proof planning from proof execution*.
5. **Unfocused verification**: Models dilute attention across heterogeneous checks → *two-stage verification (struc-*

[1]University of California, San Diego, La Jolla, USA [2]California Institute of Technology, Pasadena, USA [3]Amazon Web Services (work done outside of Amazon). Correspondence to: Chenyang An <c5an@ucsd.edu>.

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).

tural then detailed).

6. **Problem modification**: Models alter the problem statement to reduce difficulty → *problem-statement integrity checking (word-by-word comparison)*.
7. **Single-model bottleneck**: Relying on one model makes the system vulnerable to that model's blind spots → *multi-model parallel proving and brainstorm mode*.

Through our collaboration with seven domain experts, we asked experts to propose research projects for which, to the best of their knowledge, no proof or complete solution was known, and whose successful resolution would constitute a publishable contribution. We collected 18 research projects and tested QED on them. QED successfully produced original, nontrivial, and complete solutions to 5 research projects in algebraic geometry, fluid PDE, probability, and inverse problems. QED was operated fully automatically, and no expert guidance was provided during the runs. Each proof was independently verified by the contributing domain experts, who confirmed the results to be *original* (not previously known) and *nontrivial* (requiring genuine mathematical insight). Moreover, the observed false positive rate of the QED verifier with Codex GPT-5.5 in our experiment was 0 out of 17. In particular, across 214 proof candidates produced by the QED prover using Codex GPT-5.5 and evaluated by the QED verifier under the same configuration, 17 candidates were accepted by the verifier. Every verifier-accepted candidate was subsequently accepted by the corresponding domain expert, yielding an observed false-positive rate of 0 in this setting.

Links to the problem statements, proofs, and expert commentary for the five research projects that QED successfully solved can be found in the appendix A.

Note that we do not report full details of all the unsuccessful cases for QED because some of the research problems remain active, with human mathematicians still working on them. Many of the unsuccessful cases are longstanding open problems that are widely regarded as genuinely difficult, and we share some of them in the appendix B.

In summary, our contributions are as follows:

- We give an empirical characterization of seven failure modes of frontier LLMs on research-level proof tasks (Section 3).
- We design and release **QED**, an open-source multi-agent proof system that fully automates the theorem-proving process given human-provided problem statements, with each architectural decision driven by a specific failure mode (Section 4).
- Through collaborations with domain experts, we show that QED independently produced original, nontrivial proofs for five open research projects across probability, algebraic geometry, fluid PDEs, and inverse problems, with all resulting proofs verified and validated by the relevant experts. Expert assessments further indicate that three of these results are comparable in difficulty and scope to work published in established specialist mathematics journals. We also report selected unsuccessful cases, providing direct evidence of both the capabilities and current limitations of LLMs when embedded in a well-designed multi-agent pipeline for open mathematical research problems.

2. Related Work

LLM mathematical reasoning. The rapid improvement of LLMs on open-source mathematical benchmarks, from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) to competition-level problems (Trinh et al., 2024), has made those benchmarks unable to measure SOTA models' reasoning capabilities. Chain-of-thought prompting (Wei et al., 2023), self-consistency (Wang et al., 2023), training models on better data (An et al., 2024; 2025a), and reinforcement learning from human feedback or reward models (Guo et al., 2025; An et al., 2025b) have all contributed to this progress. However, benchmark performance does not directly imply the ability to produce original research-level proofs, as benchmarks consist of problems with known solutions. To address the saturation of open benchmarks, more recent efforts have introduced harder, often closed-source evaluations such as Frontier Math and the Humanity's Last Exam benchmark, which aim to probe the limits of current models on substantially more challenging and less memorized problems (Phan et al., 2025; Glazer et al., 2025). However, these benchmarks are not publicly available, limiting reproducibility and systematic analysis.

Agent architectures for coding and scientific discovery. Recent multi-agent LLM systems increasingly move beyond simple chat-based collaboration toward structured agent architectures for complex technical and scientific workflows. In software engineering, systems such as MAAD (Li et al., 2025) and ALMAS (Tawosi et al., 2025) decompose development into specialized roles and stages, coordinating agents for architecture design, implementation, validation, and integration. In scientific discovery, systems such as MatSciAgent (Chaudhari et al., 2026), Robin (Ghareeb et al., 2025), DORA AI Scientist (Naumov et al., 2025), and MACC (Oyama et al., 2026) organize agents around domain-specific scientific workflows, including hypothesis generation, literature analysis, experiment design, tool use, and report generation. More broadly, CORAL (Qu et al., 2026) studies autonomous multi-agent evolution for open-ended discovery, emphasizing sustained search and knowledge accumulation across interacting agents (Qu et al., 2026). QED

differs from these systems in its specific focus on research-level mathematical proof, its failure-mode-driven design, and its multi-stage verification pipeline tailored to mathematical reasoning. In particular, QED explicitly targets challenges that are largely absent or less central in prior agent systems: rigorous citation grounding, preservation of problem-statement integrity, and formal consistency across a subgoal tree. Rather than relying only on role specialization, conversational refinement, or experimental feedback, QED introduces structured verification stages that enforce correctness at each level of the proof hierarchy, aligning the agent architecture with the logical and compositional nature of mathematical proof.

AI and open problems. Recent work has explored whether AI systems can contribute to open mathematical questions. FunSearch (Romera-Paredes et al., 2024) used LLMs to discover new constructions in extremal combinatorics. Aletheia (Feng et al., 2026) demonstrated a closed-source agent that made research progress on open problems across multiple mathematical domains. OpenAI has also reported progress on Erdős-style open problems: two manuscripts present eight short proofs across combinatorics, number theory, and probability, with the authors attributing the proofs to internal OpenAI models and describing the human role as digesting and polishing the write-ups (Alexeev et al., 2026b;c). Its recent planar unit-distance result similarly reports a counterexample to a longstanding conjecture produced by an internal reasoning model (OpenAI, 2026). Google DeepMind's AI Co-Mathematician is a limited-release Gemini-based workbench in which a project coordinator agent manages a stateful mathematical workspace and delegates tasks across parallel workstreams and specialized agents; the system is designed to support open-ended research workflows such as ideation, literature search, computational exploration, theorem proving, and theory building, and reports early tests in which it helped researchers solve open problems, identify new directions, and uncover overlooked references (Zheng et al., 2026). However, these results are either produced by closed-source agent or have faced scrutiny: some attributed solutions to open problems (e.g., those associated with Erdős) were argued to be rediscoveries, and construction-focused results do not constitute proofs in the traditional sense. QED contributes to narrowing this gap as an open-source system producing natural-language proofs for genuinely open research problems, verified as original by domain experts.

3. Why LLMs Fail at Real Proofs

Before introducing the architecture of QED, we characterize the failure modes that motivated its design. These observations emerge from experimentation with frontier coding agents (Claude Code, Codex and Gemini CLI) on research-level mathematical problems, prior to and during the development of QED.

3.1. Failure Mode 1: Context Contamination

When a single LLM session serves as both prover and verifier, the verification step is contaminated by the prover's reasoning context. The model "remembers" its own proof strategy and is predisposed to accept its own arguments, even when they contain errors. This produces a false sense of correctness: the model generates a flawed proof, then verifies it as correct because both steps share the same internal state.

In our earliest attempts, we used the same LLM model to both generate and verify proofs, although in separate calls. This avoided literal context leakage, but still suffered from same-model self-verification bias: the verifier shared the prover's reasoning tendencies and often failed to identify subtle flaws in proofs generated by that same model. This led to more frequent false positives and motivated our later design with a more independent verification stage.

3.2. Failure Mode 2: Citation Hallucination

LLMs routinely fabricate mathematical citations. They invent theorem numbers, attribute results to wrong sources, fabricate URLs, or cite statements that do not appear in the referenced source. This is particularly dangerous in proof generation because a proof that relies on a hallucinated lemma may appear logically sound while being built on a nonexistent foundation.

In our experiments, citation hallucination was not a rare edge case. When no dedicated citation verifier was used, we observed that a high percentage of the citations produced by strong models such as Claude 4.6 Opus and GPT-5.5 in research-level proofs contained some form of hallucination or grounding error. These errors included fabricated theorem statements, incorrect attributions, nonexistent URLs, and citations to sources that did not actually contain the claimed statement. This made citation verification essential: without it, a proof could appear mathematically coherent while relying on unsupported or nonexistent lemmas.

3.3. Failure Mode 3: Misallocation of Proof Effort

In our experiments, we found that LLMs tend to evade the core mathematical difficulties: while they often recognize what needs to be proven, they rely on vague language to bypass the rigorous execution required for genuine insight, resorting to phrases like "clearly, X holds" or "it is straightforward to verify" when the claim is neither clear nor straightforward. Meanwhile, they overinvest attention in elementary components, providing excessive detail to reprove well-established lemmas or justify routine steps; by exhaust-

ing their focus on these trivialities, they leave the primary, novel complexities of the problem largely underexplored.

3.4. Failure Mode 4: Unstable Proof Plans

In our experiments, we found that LLMs typically form a proof plan before executing it, but they give up on plans too easily. When a model encounters a difficult step during execution, it tends to abandon the current plan and switch to an entirely different strategy rather than pushing through the difficulty. Over multiple iterations, this produces a random walk through proof space rather than systematic progress: the model cycles through superficially different plans without ever committing to one long enough to resolve its hard steps. The model fails to distinguish between "this approach has a fixable error in execution" and "this approach is fundamentally flawed and needs a new plan."

3.5. Failure Mode 5: Unfocused Verification

When a single verification agent is tasked with checking everything—problem-statement integrity, citations, logical structure, and step-by-step correctness—it tends to dilute its attention across all aspects and miss critical issues. This has been identified repeatedly in our experiments.

3.6. Failure Mode 6: Problem Modification

We also observed a form of reward hacking in proof generation: models sometimes subtly changed the problem being solved while still producing a plausible proof. For example, the model might change a quantifier, strengthen an assumption, weaken the conclusion, restrict the domain, or prove only a special case while presenting it as the original theorem. This is dangerous because the resulting proof may be mathematically correct, but for the wrong problem. In this sense, the model optimizes for the verifier's proxy objective—producing something that looks like a valid proof—rather than satisfying the true objective of proving the exact user-specified statement. This is closely related to reward hacking or specification gaming, where a model exploits imperfections in the evaluation criterion to achieve a high score without accomplishing the intended task (Bondarenko et al., 2025; Wang et al., 2026).

3.7. Failure Mode 7: Single-Model Bottleneck

Different LLMs have different mathematical strengths and blind spots. A problem that one model cannot solve may be accessible to another. Relying on a single model means the system inherits all of that model's weaknesses, with no mechanism for complementary coverage.

4. QED System Design

QED is a multi-stage, multi-agent pipeline for mathematical proof generation. Its architecture is directly motivated by the failure modes identified in Section 3: each design decision addresses a specific observed weakness. Figure 1 provides an overview. We describe QED's two proof modes (simple and decomposition), its verification pipeline, and its multi-model capabilities.

4.1. Simple Mode

QED simple mode (as shown in Fig 1) implements a three-stage pipeline, where Stage 0 performs literature survey and Stage 2 summarizes the entire proof effort. Stage 1 is essentially an iterative search–verify loop, suitable for problems that can be solved through progressive refinement.

Per-round flow in stage 1 In each round $r = 1, \ldots, R_{\max}$, the system does the following steps:

1. **Proof search**: One or more proof agents write a complete proof, informed by the literature survey. For rounds $r > 1$, the prover receives the previous round's proof and its verification results, so it can learn from past failures and attempt different strategies.
2. **Verification**: Each proof is independently checked by structural verification (Phases 1–5: problem-statement integrity, completeness, citations, subgoal tree structure, and any human-specified rules) followed by detailed verification (Phase 6: step-by-step logical and mathematical checking).
3. **Selection**: When multiple proofs are generated, a selector agent evaluates all proof–verification pairs and selects the best candidate.
4. **Verdict**: A verdict agent reads the selected proof and its verification reports and decides DONE (proof accepted) or CONTINUE (next round). Note that the verdict agent does not read the proof or generate any new verification reports.

Addressing Failure Mode 1 (context contamination). The proof agent and verification agents are separate model invocations with no shared context. The verifier sees only the proof document and the original problem—not the prover's reasoning process or intermediate attempts.

Addressing Failure Mode 7 (single-model bottleneck). When multi-model mode is enabled, multiple coding agents (e.g., Claude Code, Codex, Gemini CLI) generate proofs in parallel. Each proof is then independently verified by one or more verification agents, which may themselves use different models. A selector agent evaluates all proof–verification pairs and selects the best candidate. An optional

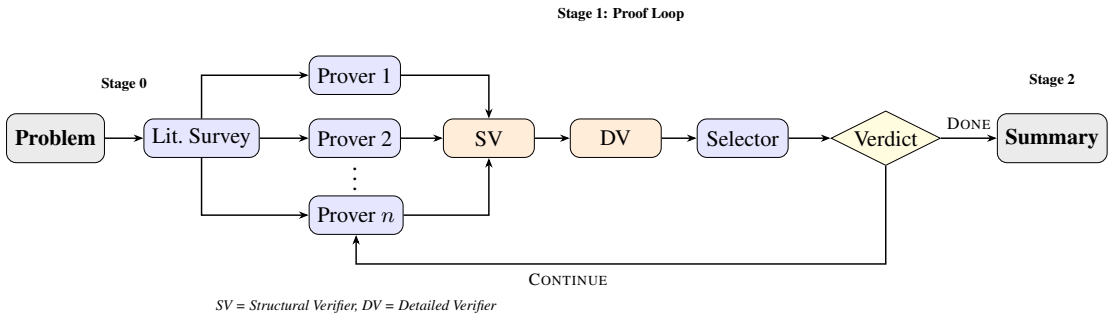

Figure 1. QED system architecture. Stage 0 surveys the literature. Stage 1 iterates a proof loop: multiple models generate proofs in parallel; the structural verifier (SV) and detailed verifier (DV) each independently produce $n \times m$ verification reports (m per proof), where n is the number of provers and m is the number of verification agents for each proof. At present, both n and m are at most three, corresponding to Claude Code, Codex, and Gemini CLI. A selector reads all verification results and picks the best proof with its verification result; and a verdict agent makes the final accept/continue decision based on the verification result of the selected proof. Stage 2 summarizes the entire proof effort. The prover can also access the internet when additional literature review is required.

brainstorm step allows multiple models to propose proof strategies before the main search begins.

4.2. Decomposition Mode

For harder problems, QED offers a decomposition mode that separates proof planning from proof execution, directly addressing Failure Mode 4 (unstable proof plans).

Agents. Decomposition mode employs six specialized agents:

1. **Decomposer**: Creates a structured proof plan—a directed acyclic graph (DAG) of intermediate mathematical claims with explicit dependencies. It has access to previous plans and regulator output that identifies the failure reasons for the previous plans, allowing it to avoid repeating them or to revise them.
2. **Single prover**: Writes a complete proof following the decomposer's plan. When the regulator decides REVISE_PROOF, the prover receives the previous proof and its verification feedback to guide revision.
3. **Structural verifier**: Phases 1–5 are the same as in the simple mode, except for an added criterion that checks whether the proof produced by the prover follows the decomposition plan.
4. **Detailed verifier**: Phase 6 (same as the verification step in simple mode).
5. **Regulator**: Analyzes verification failures and decides the next action.
6. **Verdict**: Final accept/reject decision, same as the verdict agent in the simple mode.

Three-level retry hierarchy. The regulator's key innovation is distinguishing three types of failure, each triggering a different corrective action:

Decision	Diagnosis	Action
REVISE_PROOF	Execution errors, plan is sound	Re-prove with same plan
REVISE_PLAN	Structural gaps in the plan	Modify plan, then re-prove
REWRITE	Fundamental approach is flawed	New decomposition from scratch

Table 1. Three-level retry hierarchy in decomposition mode. The regulator distinguishes execution errors from plan-level and strategy-level failures. The decomposer-regulator design addresses Failure Mode 4.

Each level has a configurable limit (e.g., 3 proof attempts per plan revision, 2 plan revisions per decomposition, 3 total decompositions), preventing infinite loops while allowing systematic exploration.

Proof plan structure. The decomposer outputs a YAML-formatted plan where each step is a precise quantitative mathematical statement (not a vague description like "show the function grows slowly"). Each step specifies its dependencies, difficulty level, and whether it is a *key step*—the most novel and difficult part of the proof. The plan also includes cited sources and a mandatory self-critique section.

4.3. Verification Pipeline

QED's verification pipeline consists of six phases, split into two stages. This two-stage design addresses Failure Mode 5 (unfocused verification) by allowing each stage to focus on a specific class of checks.

Stage A: Structural Verification (Phases 1–5). Structural verification checks the proof's foundations before investing in expensive step-by-step analysis:

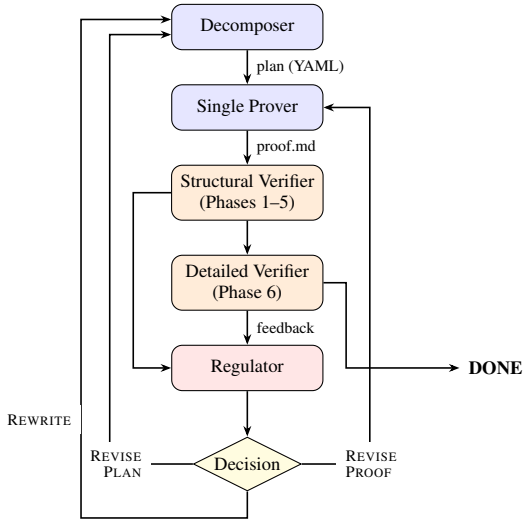

Figure 2. Decomposition mode flow. The regulator analyzes verification failures and selects one of three corrective actions: REVISE_PROOF revises the proof without changing the plan, REVISE_PLAN modifies the existing decomposition plan, and REWRITE works on a new decomposition entirely. The decomposer has access to previous plans to avoid repetition or make revisions.

- **Phase 1: Problem-statement integrity** (addresses Failure Mode 6). The verifier reads the original problem verbatim and compares it word-by-word against the claim the proof actually proves. *Any* discrepancy—changed quantifiers, strengthened hypotheses, weakened conclusions, restricted domains, swapped conclusions and hypotheses—is an automatic failure.
- **Phase 2: Completeness and originality.** Checks that all parts of the problem are addressed and that the proof contains genuine argument (not just a literature survey or proof sketch).
- **Phase 3: Citation verification** (addresses Failure Mode 2). For every citation the verifier independently checks: (a) the source URL resolves, (b) title and authors match, (c) the exact statement exists at the specified location, (d) the cited statement matches the source verbatim, and (e) the proof correctly applies the cited result with all conditions satisfied.
- **Phase 4: Subgoal tree validation.** The proof must declare its logical architecture as a tree of subgoals rooted at the main claim. The verifier checks well-formedness (no orphans, no cycles), validity of each reduction, and completeness (every subgoal resolved, every cited theorem's conditions accounted for as explicit subgoals).
- **Phase 5: Human-specified rules.** If domain experts have provided additional verification criteria, these are checked as hard requirements.

Stage B: Detailed Verification (Phase 6). Phase 6 runs *only if* Phases 1–5 pass. It performs step-by-step logical and mathematical checking:

- For each assertion: state the claim, quote the justification, list dependencies, check logical validity, verify computations, and assign a verdict (Pass/Fail/Uncertain).
- Cross-reference with citation verdicts from Phase 3.
- Verify all subgoal resolutions point to specific, real parts of the proof.
- Analyze `<key-original-step>` tags (addresses Failure Mode 3): check that every tagged step is genuinely nontrivial and maximally detailed, and that every nontrivial step is properly tagged.

4.4. Prover Output Format Enforcement

In both simple mode and decomposition mode, QED imposes a structured output format on the prover to address multiple failure modes simultaneously. To address Failure Mode 2 (citation hallucination), the prover must use a standardized citation format: every cited result must include the exact theorem statement, the source (with title, authors, and location), and a verification that all conditions of the cited result are satisfied in the current context. To address Failure Mode 3, QED requires the prover to wrap every nontrivial original step with `<key-original-step>` tags. Content inside these tags must be "maximally detailed"—every sub-step, every bound, every case made explicit. The verifier then checks:

- Every nontrivial proof has at least one key-original-step.
- Tagged content contains no vague language ("clearly," "obviously," "straightforward").
- Untagged nontrivial steps are flagged as potential hidden weaknesses.
- Routine steps inflated with key-original tags are flagged.

This creates a forcing function: the prover must *identify* the hardest parts of the proof and provide complete detail for exactly those parts.

4.5. Optional Human-in-the-Loop Steering

While QED runs fully automatically by default, it supports optional human steering between rounds. Domain experts can provide:

- **Global proof hints**: Persistent guidance across all rounds (e.g., "try Fourier-analytic methods").
- **Additional verification rules**: Extra criteria the verifier should enforce.

These are read from designated files before each new round begins. In our evaluation on open research problems (Section 5), this feature was *not used*—all results were obtained without expert guidance.

4.6. Standalone Proof Verifier

In addition to the full proof-generation pipeline, QED provides a *standalone proof verifier* that takes any problem–proof pair and produces a structured verification report. This tool is designed for users who already have a proof (whether human-written or AI-generated) and want to check its correctness without running the full proof-search loop.

The standalone verifier uses a difficulty-adaptive architecture with up to three agents:

1. **Difficulty judge.** A judge agent classifies the problem as *Easy* or *Hard*. For easy problems, the judge performs a complete streamlined verification in a single pass and the pipeline terminates. For hard problems, the judge defers to two specialized verification agents.
2. **Structural verifier** (hard only). Checks problem-proof alignment, completeness and originality, and the proof's logical architecture.
3. **Detailed verifier** (hard only, after structural pass). Performs step-by-step logical verification, mathematical correctness checks, rigor analysis, and coverage checks. Skipped if structural verification fails.

4.7. Automatic Resume

QED supports automatic resume from any point in the pipeline. If the process exits for any reason—model API failures, network interruptions, machine restarts, or resource limits—QED detects the exact stage of progress by scanning the output directory structure and resumes from where it left off. This applies to both simple and decomposition modes: in simple mode, QED can resume at any sub-stage within any round; in decomposition mode, it restores the full attempt–revision–proof hierarchy and continues from the last incomplete step. This is essential for research-level proof tasks, which may run for extended periods across many rounds.

5. Experiments

We evaluate QED on 18 open research projects contributed by domain experts, with no proof known for any of the problems in those projects. Our evaluation takes the form of case studies rather than benchmark experiments. We do not report full details of all the unsuccessful cases for QED because some of the research problems remain active, with human mathematicians still working on them. We also do not include full ablation studies over all projects due to two considerations. First, no benchmark exists for open research problems. By definition, these problems lack known solutions against which to measure performance. Second, verifying the correctness and originality of a proof requires substantial effort from domain experts. Running QED with each component disabled would multiply the number of proofs requiring expert review, imposing an impractical burden on the mathematicians who volunteered their time. Instead, Section 6 reports targeted diagnostic ablations on the two probability cases where repeated system variants could be compared without requiring broad expert re-review. In practice, for each non-probability problem, we asked the corresponding expert to assess the correctness of QED's proof at most once; if the proof was found to be incorrect, we did not repeatedly revise and resubmit proofs for the same problem, in order to respect the experts' limited availability and avoid overburdening them.

5.1. Experiment Setup

Research Projects. 18 research projects were contributed by 7 (groups of) experts. Each project may contain 1 or more open problems.

- **Projects 1–10**: Contributed by a domain expert in probability. Some of the problems in the projects arose from domain expert's active research, and some of them are well-known, genuinely difficult open problems in the area.
- **Projects 11–12**: Contributed by a domain expert in operator algebra. Both of them are well-known, genuinely difficult open problems in the area.
- **Projects 13–14**: Contributed by a domain expert in algebraic geometry. All problems in the projects arose from domain expert's active research.
- **Projects 15–17**: Contributed by a domain expert in fluid PDE. All problems in the projects arose from domain expert's active research.
- **Project 18**: Contributed by a group of domain experts in inverse problems, concerning Carleman weight functions for PDEs. The statement and the proof are withheld currently.

The 5 research projects that QED successfully solved independently are documented in Appendices A.1–A.5. They correspond respectively to Project 6 (probability), Project 8 (probability), Project 13 (algebraic geometry), Project 15 (fluid PDE), and Project 18 (inverse problems). Selected unsuccessful projects are presented in Appendix B.

Configuration and Costs. The model configurations differed by projects:

- **A.1** Decomposition mode. All agents use Codex with GPT-5.5-pro, thinking level xhigh. This case costs around 1000 US dollars.
- **A.2–A.3** Decomposition mode. All agents use Codex with GPT-5.5, thinking level xhigh. A.2 costs around 200 US dollars. A.3 costs around 100 US dollars.
- **A.4** There are three open research problems in this case. The first problem is completed with simple mode, with all agents using Codex with GPT-5.4, thinking level xhigh. The second and third problems are completed with decomposition mode, with all agents using Codex with GPT-5.5, thinking level xhigh. This case costs around 600 US dollars.
- **A.5** Simple mode. Literature survey, proof search, all verifications and verdict use Anthropic Claude Opus 4.6, brainstorm is not enabled. This case costs around 50 US dollars.

We report the retry details for C1–C5, including the number of plan rewrites, plan revisions, and proof revisions, in Appendix C.

Independence. The system received only the problem statements and configuration parameters; no domain-specific guidance or expert feedback was provided.

5.2. Results

Expert Assessment of Significance To assess the mathematical significance of the results produced by QED, we asked the collaborating domain experts to evaluate the completed solutions in the context of their respective fields. The experts confirmed that the resulting proofs are correct, original, and nontrivial. They further characterized the significance of the individual contributions as follows: A.1 was described as comparable to work published in the *Electronic Journal of Probability* or the *Proceedings of the American Mathematical Society*; A.2 was characterized as PhD-level work with nontrivial technical depth; the proof in A.3 was described as original, nontrivial, elementary, elegant, and "moderately difficult"; A.4 was described as comparable to work published in venues such as the *Journal of Differential Equations* or *SIAM Journal on Mathematical Analysis*; and A.5 was characterized as comparable to work published in journals such as the *Journal of Mathematical Analysis and Applications*. These assessments suggest that QED is not merely generating formal exercises or routine derivations, but is capable of producing solutions in diverse math domains, whose mathematical content reaches the level of publishable research contributions.

See appendix A for the statements, proof links, and full expert comments.

Verifier Reliability In the QED pipeline, each proof candidate generated by the QED prover is checked by the QED verifier. We analyze the subset of QED runs in which the verifier used the Codex GPT-5.5 configuration. This subset contains 214 proof candidates spanning all 18 research projects considered in this work.

Among these 214 proof candidates verified by the QED verifier with Codex GPT-5.5, 17 were accepted by the verifier. Each of these 17 verifier-accepted proofs was subsequently reviewed and accepted by the corresponding domain expert. Thus, within the proof candidates checked under this verifier configuration, we observed no false positives: there was no case in which the verifier accepted a proof candidate that was later rejected by expert review.

6. Discussion

Beyond benchmarks. These results demonstrate a capability qualitatively different from the benchmark performance discussed in Section 2. Solving a competition problem with a known solution is fundamentally different from producing an original proof for a problem whose answer is not known in advance, which is why the evaluation in Section 5 is organized as expert-assessed research case studies rather than as benchmark scoring.

Originality. A central concern about AI-generated mathematics is whether the resulting proofs are genuinely original. We argue that the proofs produced by QED for the five research projects above satisfy this criterion. The problems addressed by QED were open research questions with no known solutions, as described in Section 5; they were newly proposed by domain experts in the course of their own recent research. QED produced all proofs without domain-expert guidance during the proof-generation process, consistent with the independence protocol in Section 5 and the unused human-steering option in Section 4.5. For Appendices A.1–A.5, the corresponding experts characterized QED's contributions as original.

Targeted ablation studies. Full ablations are impractical for the reasons described in Section 5: each candidate research proof requires time-intensive expert review. We therefore performed targeted qualitative ablations on the two probability cases in Appendices A.1 and A.2. These two cases were selected because both were solved successfully by QED and expose the failure modes that motivated the system design in Sections 3 and 4. As a single-query baseline, we submitted each problem to GPT-5.5 Pro in five independent attempts. None of the ten attempts produced a complete proof; all omitted essential steps. We then inspected variants in which specific QED constraints were removed. The goal of these ablations is not to estimate a suc-

cess rate, but to test whether individual system components prevent the concrete failure modes identified in Sections 3.2 to 3.4.

- **Standardized citation format.** Without the format required in Section 4.4, the prover cited results without stating them precisely. In some attempts, it misstated a cited theorem, making the claimed application invalid. In others, the application was correct but insufficiently explained, falling short of mathematical writing standards and making the argument difficult for human experts to verify. This supports the claim from Sections 3.2 and 4.4 that citation structure is not merely cosmetic: it forces cited results to be explicit enough for both verifier checking and expert review.
- **Citation verification (Phase 3).** Without independent citation verification, one attempt used a theorem found only in unpublished research notes. As a result, a key step in the proof relied on a less trustworthy source that readers could not independently verify. This supports the role of Phase 3 in Section 4.3 and directly addresses the citation hallucination failure mode in Section 3.2: the verifier must check not only whether a statement is mathematically plausible, but whether the cited source is public, precise, and usable in the proof.
- **`<key-original-step>` tags.** Both probability questions require matching lower and upper bounds. Without mandatory tagging, the prover developed the easier lower bound but left the harder upper bound as a sketch. This illustrates the misallocation of proof effort described in Section 3.3. It also supports the purpose of `<key-original-step>` tags in Section 4.4 and their verification role in Section 4.3: they force the prover and verifier to identify the mathematically novel step and demand maximal detail precisely there.
- **Decomposition mode.** Neither question was solved completely in simple mode, whose iterative search–verify loop is described in Section 4.1. Across repeated attempts, the prover frequently changed its overall proof strategy, indicating that the proof plan was unstable in the sense of Section 3.4. As a result, no single approach was developed in sufficient depth. The manuscripts produced in simple mode were at most 10 pages long. By contrast, decomposition mode produced the expert-accepted proofs in Appendices A.1 and A.2, which were 20 and 15 pages, respectively. Although length alone does not measure proof quality, the qualitative difference was that decomposition mode preserved a stable plan long enough to resolve the difficult upper-bound arguments. This supports the claim in Section 4.2 and Table 1 that separating planning from execution helps address unstable proof plans.

These ablations are limited to two cases and are diagnostic rather than statistically powered. Nevertheless, they show that the tested components address distinct, directly observed failure modes from Section 3: citation grounding, verification of external dependencies, allocation of effort to key original steps, and stability of the proof plan.

7. Conclusion

We presented QED, an open-source multi-agent system that fully automates theorem proving given human-provided questions. Through collaborations with domain experts, QED produced original and nontrivial proofs for five research projects across probability, algebraic geometry, fluid PDEs, and inverse problems, with three of them comparable in difficulty and scope to work commonly published in established specialist mathematics venues. In each successful case, the system received only the problem statement and configuration parameters, ran without human guidance during proof generation, and produced outputs subsequently validated by the relevant experts. These results provide evidence that QED has a meaningful success rate for producing original and nontrivial proofs in mathematical research.

Acknowledgments

We thank Xiaoqian Xu (Assistant Professor of Mathematics at Duke Kunshan University and Zu Chongzhi Center for Mathematics and Computational Sciences) for contributing Projects 15–17 and providing expert verification and commentary. We thank Yilong Zhang (Limited Term Assistant Professor in the Department of Mathematics at the University of Georgia) for contributing Projects 13–14 and providing expert verification and commentary. We thank Qiao Zhuang (Assistant Professor in the Department of Mathematics and Statistics at University of Missouri–Kansas City), Zhongqiang Zhang (Associate Professor in the Department of Mathematical Sciences at Worcester Polytechnic Institute), and Yu Wang (Assistant Professor in the School of Mathematics at Southwest Jiaotong University) for contributing Project 18 and providing expert verification and commentary. We thank Kewei Xu (Kai Toyosawa, Postdoctoral Researcher in the Department of Mathematics, University of Münster) for contributing Projects 11–12 and providing expert verification. We thank Sam Buss (Professor at University of California, San Diego) for his continued support and insightful discussions.

References

Alexeev, B., Barreto, K., Li, Y., Lichtman, J. D., Price, L., Shah, J. I., Tang, Q., and Tao, T. Primitive sets and von mangoldt chains: Erdős problem #1196 and beyond, 2026a. URL https://arxiv.org/abs/2605.00301.

Alexeev, B., Putterman, M., Sawhney, M., Sellke, M., and Valiant, G. Short proofs in combinatorics and number theory, 2026b. URL https://arxiv.org/abs/2603.29961.

Alexeev, B., Putterman, M., Sawhney, M., Sellke, M., and Valiant, G. Short proofs in combinatorics, probability and number theory ii, 2026c. URL https://arxiv.org/abs/2604.06609.

An, C. and Pan, M. Return probability for the switch–walk–switch lamplighter walk on a regular tree, 2026. URL https://arxiv.org/abs/2605.21744.

An, C. and Xu, X. Lower bounds for advection-diffusion equations: An exploration with ai-generated proofs, 2026. URL https://arxiv.org/abs/2605.20623.

An, C., Chen, Z., Ye, Q., First, E., Peng, L., Zhang, J., Wang, Z., Lerner, S., and Shang, J. Learn from failure: Fine-tuning LLMs with trial-and-error data for intuitionistic propositional logic proving. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 776–790, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.45. URL https://aclanthology.org/2024.acl-long.45.

An, C., Imani, S., Yao, F., Dong, C., Abbasi, A., Shrivastava, H., Buss, S., Shang, J., Mahalingam, G., Sharma, P., and Diesendruck, M. Next-token prediction task assumes optimal data ordering for llm training in proof generation, 2025a. URL https://arxiv.org/abs/2411.00863.

An, C., Qin, Z., Kong, S., and Flamant, C. DeRL: Diverse-exploration reinforcement learning for large language models improves mathematical reasoning, 2025b. URL https://openreview.net/forum?id=ZIYYeTkZQ4.

Arapura, D., Greer, F., and Zhang, Y. Failure of the invariant cycle theorem over F, 2026. URL https://arxiv.org/abs/2602.07302.

Bondarenko, A., Volk, D., Volkov, D., and Ladish, J. Demonstrating specification gaming in reasoning models, 2025. URL https://arxiv.org/abs/2502.13295.

Chaudhari, A., Ock, J., and Barati Farimani, A. Modular large language model agents for multi-task computational materials science. *Communications Materials*, 2026.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Feng, T. Eigenweights for arithmetic hirzebruch proportionality, 2026. URL https://arxiv.org/abs/2601.23245.

Feng, T., Trinh, T. H., Bingham, G., Hwang, D., Chervonyi, Y., Jung, J., Lee, J., Pagano, C., hyun Kim, S., Pasqualotto, F., Gukov, S., Lee, J. N., Kim, J., Hou, K., Ghiasi, G., Tay, Y., Li, Y., Kuang, C., Liu, Y., Lin, H., Liu, E. Z., Nayakanti, N., Yang, X., Cheng, H.-T., Hassabis, D., Kavukcuoglu, K., Le, Q. V., and Luong, T. Towards autonomous mathematics research, 2026. URL https://arxiv.org/abs/2602.10177.

Ghareeb, A. E., Chang, B., Mitchener, L., Yiu, A., Szostkiewicz, C. J., Laurent, J. M., Razzak, M. T., White, A. D., Hinks, M. M., and Rodriques, S. G. Robin: A multi-agent system for automating scientific discovery, 2025. URL https://arxiv.org/abs/2505.13400.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., de Oliveira Santos, E., Järviniemi, O., Barnett, M., Sandler, R., Vrzala, M., Sevilla, J., Ren, Q., Pratt, E., Levine, L., Barkley, G., Stewart, N., Grechuk, B., Grechuk, T., Enugandla, S. V., and Wildon, M. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai, 2025. URL https://arxiv.org/abs/2411.04872.

Gowers, T. A recent experience with chatgpt 5.5 pro. Gowers's Weblog, May 2026. URL https://gowers.wordpress.com/2026/05/08/a-recent-experience-with-chatgpt-5-5-pro/. Accessed May 21, 2026.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye,

S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-R1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Li, R., Zhang, Y., Zhou, X., Liang, P., Sun, W., Xuan, J., Jin, Z., and Liu, Y. Maad: Automate software architecture design through knowledge-driven multi-agent collaboration, 2025. URL https://arxiv.org/abs/2507.21382.

Naumov, V., Zagirova, D., Lin, S., Xie, Y., Gou, W., Urban, A., Tikhonova, N., Alawi, K., Durymanov, M., Galkin, F., et al. Dora ai scientist: Multi-agent virtual research team for scientific exploration discovery and automated report generation. *bioRxiv*, 2025.

OpenAI. Planar point sets with many unit distances. OpenAI technical proof manuscript, 2026. URL https://cdn.openai.com/pdf/74c24085-19b0-4534-9c90-465b8e29ad73/unit-distance-proof.pdf. Accessed May 21, 2026.

Oyama, S., Sakurai, Y., and Kashima, H. Macc: Multi-agent collaborative competition for scientific exploration, 2026. URL https://arxiv.org/abs/2603.03780.

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., Zhang, C. B. C., Shaaban, M., Ling, J., Shi, S., et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qu, A., Zheng, H., Zhou, Z., Yan, Y., Tang, Y., Ong, S. Y., Hong, F., Zhou, K., Jiang, C., Kong, M., Zhu, J., Jiang, X., Li, S., Wu, C., Low, B. K. H., Zhao, J., and Liang, P. P. Coral: Towards autonomous multi-agent evolution for open-ended discovery, 2026. URL https://arxiv.org/abs/2604.01658.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J., Ellenberg, J. S., Wang, P., Fawzi, O., et al. Mathematical discoveries from program search with large language models. *Nature*, 625 (7995):468–475, 2024.

Tawosi, V., Ramani, K., Alamir, S., and Liu, X. Almas: an autonomous llm-based multi-agent software engineering framework, 2025. URL https://arxiv.org/abs/2510.03463.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Wang, X., Tian, M., Zeng, Y., Huang, Z., Yuan, J., Chen, B., Xu, J., Zhou, M., Liu, W., Wu, M., Guo, Z., Qian, Q., Wang, Y., Zhang, F., Yin, R., Dou, S., Lv, C., Chen, T., Song, K., Tan, X., Gui, T., Zheng, X., and Huang, X. Reward hacking in the era of large models: Mechanisms, emergent misalignment, challenges, 2026. URL https://arxiv.org/abs/2604.13602.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Zheng, D., von Glehn, I., Zwols, Y., Beloshapka, I., Buesing, L., Roy, D. M., Wattenberg, M., Georgiev, B., Schmidt, T., Cowie, A., et al. Ai co-mathematician: Accelerating mathematicians with agentic ai. *arXiv preprint arXiv:2605.06651*, 2026.

A. Successful Projects

Table 2 collects the public proof records, expert comments, and withheld items for the five successful projects.

A.1. Project 6 — Return Probability Asymptotics for the Lamplighter Walk on $\mathbb{Z}_2 \wr T_d$

Problem Statement. Let $d \geq 3$ and let T_d denote the infinite d-regular tree. Consider the switch–walk–switch lamplighter random walk on the wreath product $\mathbb{Z}_2 \wr T_d$, and

Table 2. Appendix proof index for successful QED projects.

Description	Domain	Record
Appendix A.1	Probability	Proof in (An & Pan, 2026); expert comments: https://github.com/proofQED/QED/blob/main/proved_statements/prob-May-15-2026/README.md
Appendix A.2	Probability	Proof and expert comments in https://github.com/proofQED/QED/blob/main/proved_statements/prob-May-15-2026
Appendix A.3	Algebraic geometry	Proof and expert comments in https://github.com/proofQED/QED/blob/main/proved_statements/algebraicgeometry-May-17-2026/problem-1-correct-proof.md
Appendix A.4	Fluid PDE	Proof in (An & Xu, 2026); expert comments: https://github.com/proofQED/QED/blob/main/proved_statements/analysis-May-19-2026/README.md
Appendix A.5	Inverse problems	Withheld until related work is archived.

let $p_{2n}(e,e)$ denote the return probability of this walk to the identity after $2n$ steps. Set

$$\rho_d \;=\; \frac{2\sqrt{d-1}}{d}.$$

Then the following sharp asymptotic holds:

$$p_{2n}(e,e) \;=\; \rho_d^{2n}\,\exp\Big[-\big(\pi^2(\log(d-1))^2 + o(1)\big)\,\tfrac{n}{\log^2 n}\Big].$$

Workflow. The domain expert provided the problem statement to QED with no further mathematical input. QED was run in decomposition mode and produced the correct proof through multiple rounds of refinement, including substantial changes of proof plan by the decomposition agent. The resulting proof was then verified by the expert.

Expert Comment This result is a solid specialized contribution in probability theory. Its proof has real mathematical content as it cleverly combines probabilistic constructions with spectral analysis. Its significance appears comparable to work suitable for venues such as the *Electronic Journal of Probability* or *Proceedings of the American Mathematical Society*. The question is in the style of precise probabilistic calculations and estimates.

A.2. Project 8 — Total Variation Asymptotics for the Switch–Walk–Switch Walk on $\mathbb{Z}_2 \wr \mathbb{Z}$

Problem Statement. Consider the discrete-time switch–walk–switch random walk on the wreath product $\mathbb{Z}_2 \wr \mathbb{Z}$, and for a starting point $x \in \mathbb{Z}_2 \wr \mathbb{Z}$ let P_t^x denote the law of this walk at time t. Fix

$$x \;=\; (\mathbf{0},\,0) \qquad \text{and} \qquad y \;=\; (\mathbf{0},\,2).$$

Then the total variation distance between the two laws satisfies the sharp asymptotic

$$\big\|P_t^x - P_t^y\big\|_{\mathrm{TV}} \;\asymp\; t^{-1/2},$$

where $\asymp$ denotes equality up to positive multiplicative constants independent of t.

Workflow. The domain expert provided the problem statement to QED with no further mathematical input. QED was run in decomposition mode and produced the correct proof through multiple rounds of refinement, including substantial changes of proof plan by the decomposition agent. The resulting proof was then verified by the expert.

Expert Comment This is a technically nontrivial Ph.D.-level problem in probability theory. The question again is in the style of precise probabilistic calculations and estimates.

A.3. Project 13 — Local Invariant Cycle Theorem over the Integers in Degree One

Problem Statement. Let $X \to B$ be a one-parameter family of complex projective varieties over a small disk B, and assume that the central fiber is a simple normal crossing divisor. Let X_t denote a smooth nearby fiber, and let T denote the monodromy operator acting on $H^1(X_t,\mathbb{Z})$. Then the restriction map

$$H^1(X,\mathbb{Z}) \;\longrightarrow\; H^1(X_t,\mathbb{Z})^T$$

to the monodromy-invariant submodule is surjective. The classical (rational) invariant cycle theorem of Clemens–Schmid (1970s) establishes the analogous statement after tensoring with $\mathbb{Q}$; the integral version was open. In February 2026, Arapura–Greer–Zhang showed that the integral statement fails for H^2. The result above establishes the integral statement for H^1.

Workflow. At the time QED was given this problem, the human author had already discovered a proof but had not yet released it publicly. The domain expert provided the problem statement to QED with no further mathematical input. The resulting proof was independently verified by the expert and was found to be mathematically different from the human author's proof. The human author subsequently posted their own proof to arXiv, which is included in (Arapura et al., 2026).

Expert Comment The method in the AI's proof that interprets the coboundary map $H^1(X^*) \to H^2(X, X^*)$ as the residue map is original. This actually holds in higher degrees by working with Griffiths' residue, but in degree one this interpretation is the classical residue and is particularly useful for the integral local invariant cycle theorem, since one has the key lemma which is essential in the proof. The proof is elementary and elegant: the first lemma is essentially residue theory in complex analysis, and the second is essentially reduced to Picard–Lefschetz theory for curves. It provides a different point of view on the integral version of the invariant cycle theorem in degree one. The expert classified problem difficulty as follows: easy means that the expected results can be achieved by standard methods; moderately difficult means that the expected results are plausible, but the methods are not known in advance; and difficult refers to solving open problems that have remained unresolved for years, or finding unexpected examples or counterexamples. Under this expert classification, this problem is considered moderately difficult.

A.4. Project 15 — Explicit Lower Bounds for Advection–Diffusion Equations

Problem Statement. Consider the advection–diffusion equation

$$\partial_t \rho \;+\; \mathbf{u} \cdot \nabla \rho \;=\; \nu \, \Delta \rho$$

on the two-dimensional torus $\mathbb{T}^2$ with mean-zero smooth initial data $\rho_0 \not\equiv 0$ and divergence-free velocity field $\mathbf{u}$. The paper establishes explicit, constructive lower bounds on suitable norms of the solution in three settings.

Result 1 (Inviscid shear; polynomial $\dot{H}^{-1}$ lower bound). Let θ solve the transport equation

$$\partial_t \theta \;+\; U(y,t)\, \partial_x \theta \;=\; 0, \qquad \theta(0) = \theta_0,$$

with $U \in L^\infty_t W^{1,1}_y(\mathbb{T})$ and $\theta_0 \in C^\infty(\mathbb{T}^2)$ mean-zero. Then there exists a constant $c_* > 0$, depending only on θ_0 and U, such that

$$\|\theta(t)\|_{\dot{H}^{-1}} \;\geq\; \frac{c_*}{1+t^2} \qquad \text{for all } t \geq 0.$$

Result 2 (Diffusive shear; exponential L^2 lower bound and uniform mixing scale). Let ρ solve

$$\partial_t \rho \;+\; U(t,y)\, \partial_x \rho \;=\; \nu \, \Delta \rho,$$

with $U \in L^\infty_{t,y}(\mathbb{T})$ and $\nu > 0$. Then there are explicit constants $c_2, c_* > 0$, computable from ρ_0, U, and ν, such that

$$\|\rho(t)\|_{L^2} \;\geq\; \|\rho_0\|_{L^2}\, e^{-c_2 t} \qquad (t \geq 0),$$

and, in the small-diffusivity regime $0 < \nu \ll 1$, the mixing scale is uniformly bounded below:

$$\frac{\|\rho(t)\|_{\dot{H}^{-1}}}{\|\rho(t)\|_{L^2}} \;\geq\; c_* \qquad \text{for all } t \geq 0.$$

Result 3 (Fast time-periodic flows; exponential L^2 lower bound). Let $u \in L^\infty_t W^{1,\infty}_{x,y}$ be divergence-free and L-periodic in time, and let ρ solve

$$\partial_t \rho \;+\; u(At,x,y) \cdot \nabla \rho \;=\; \nu \, \Delta \rho.$$

Then there exist an explicit threshold $A_0 > 0$, exponent $c_A > 0$, and constant $C > 0$, all expressed in terms of ρ_0, u, L, and ν, such that for every $A > A_0$,

$$\|\rho(t)\|_{L^2} \;\geq\; C\, e^{-c_A t} \qquad (t \geq 0).$$

In particular, for time-independent flows and time-periodic flows of sufficiently high frequency, superexponential L^2 decay is impossible.

Workflow. The domain expert formulated the problem statements but did not intervene in the construction of the arguments; no PDE-specific guidance was given during the proof-generation process. QED produced the complete chain of reasoning, from the reduction of each problem to its step-by-step estimates, and verified the logical consistency of the derivations. The expert confirmed that the QED outputs are correct.

A.5. Project 18 — Carleman Estimate Weight Functions

Problem statement. Construct a smooth space–time weight function $\psi(x,t)$ (with auxiliary parameters) for the wave operator on a half-infinite domain such that it satisfies strict pseudoconvexity-type inequalities, positivity conditions, and sufficient growth at infinity to establish a valid Carleman estimate. *Note: the full problem statement and proof are withheld until the corresponding mathematics paper is posted on arXiv.*

Result. Using Claude Opus 4.6 for both proof generation and verification, QED (with simple mode) produced a candidate weight function and a complete proof in **Round 1**. The proof was produced without any domain-expert guidance except for the correction to the statement of problem 5 prompted by the AI's solution to the originally stated problem. Subsequent expert review determined that this solution corresponded to a degenerate case, revealing that the original constraints were too weak (a formulation issue rather than a QED failure). After strengthening the constraints, the experts instructed QED to continue from its previous candidate. QED then returned a refined weight function together with complete proofs satisfying all strengthened conditions. No human intervention was required beyond refinement of the problem specification.

Expert assessment. One of the most promising applications of QED to the theory of Carleman estimates is the automated search for admissible Carleman weights. The

success of a Carleman estimate hinges on finding a weight function finely adapted to the operator, the geometry, the boundary structure, and the relevant pseudoconvexity conditions. Traditionally, this has been a highly problem-specific, experience-driven process, involving substantial trial and error and lengthy inequality verification. A specialized QED-style system could transform this step into a modular, parallelizable, and rigorously verifiable search procedure: candidate weights are generated, tested against the required structural conditions, iteratively refined, and ultimately checked with full rigor. Such a tool would be valuable not only for accelerating well-established constructions, but also for exploring problems where the correct ansatz has yet to emerge.

Experts believe that QED's contribution to this work is comparable to that of research published in journals such as the Journal of Mathematical Analysis and Applications.

B. Unsuccessful Cases

The following problems were attempted by QED but were not solved within the system's preset budget. Table 3 gives a compact index, followed by the full problem statements retained for future evaluation.

B.1. Project 2

Problem statement. Let $d \geq 3$. Let T_d be the d-regular tree. There is a unique $\mathrm{Aut}(T_d)$-invariant probability measure μ on the set of proper d-colorings of $E(T_d)$, the set of edges of T_d.

This measure is easy to construct. Let o be a fixed root. Among the edges in T_d adjacent to o, choose the colors of the edges uniformly at random such that they are distinct. Then, working outwards independently, where every time there is a choice of $d-1$ distinct colors for $d-1$ edges to guarantee that the coloring is proper, choose the colors uniformly at random across the edges.

Prove that μ is a factor of i.i.d. Relevant definitions and properties can be found in Russell Lyons's *Factors of IID on Trees*.

B.2. Project 3

Problem statement. Let T_d be the d-regular tree for $d \geq 3$, with vertex set $V = V(T_d)$. We consider processes that are invariant under the full automorphism group $\mathrm{Aut}(T_d)$. Let S be a finite state space (such as $\{0,1\}$). For $K \subseteq V$, write $\sigma(K)$ for the σ-field on S^V generated by the coordinate projections π_x for $x \in K$.

Prove that if μ is an $\mathrm{Aut}(T_d)$-invariant, finitely dependent probability measure on S^V (for $d \geq 3$), then μ is a factor of i.i.d. The definitions of factor of i.i.d. and finitely dependent are as below, which can all be found in (arXiv:1401.4197, pp. 3–4).

Definition B.1 (FIID)**.** Let L be Lebesgue measure on $[0,1]$. We equip the domain space $[0,1]^V$ with the product measure L^V. The elements $\omega \in [0,1]^V$ of the domain space are sometimes called *labels*.

A measurable map $\phi\colon [0,1]^V \to S^V$ is called an $\mathrm{Aut}(T_d)$-**factor** if it is $\mathrm{Aut}(T_d)$-equivariant: for all $\gamma \in \mathrm{Aut}(T_d)$,

$$\phi(\gamma\omega) = \gamma\big(\phi(\omega)\big) \quad \text{for } L^V\text{-a.e. } \omega \in [0,1]^V.$$

Since the domain space is equipped with a product measure, or i.i.d., such a ϕ is called a **factor of i.i.d.**, or **FIID** for short. The push-forward measure $\phi_* L^V$ on S^V is also called an FIID.

Definition B.2 (Finite Dependence)**.** An $\mathrm{Aut}(T_d)$-invariant probability measure μ on S^V is m-**dependent** if $\sigma(K_1), \ldots, \sigma(K_p)$ are independent whenever the sets $K_1, \ldots, K_p \subseteq V$ are pairwise separated by graph distance strictly greater than m. We say that μ is **finitely dependent** if it is m-dependent for some $m < \infty$.

B.3. Project 7

Problem statement. For each n, let $(X_i)_{0 \leq i \leq n}$ be the simple nearest-neighbor random walk on $\mathbb{Z}^k$ conditional on $X_0 = 0$ and no self-intersections up to time n — that is, a self-avoiding walk. Let $d_k(n) := \mathbb{E}(\|X_n\|)$, the expected Euclidean distance of X_n from the origin. Is it true that $d_3(n) \gg n^{1/2}$?

B.4. Project 10

Problem statement. Let $T_4 = (V, E)$ be the infinite 4-regular tree. Let $(\sigma_0(v))_{v \in V}$ be i.i.d. with

$$\mathbb{P}_p(\sigma_0(v) = +1) = p, \qquad \mathbb{P}_p(\sigma_0(v) = -1) = 1 - p.$$

Run continuous-time zero-temperature Glauber dynamics: each vertex has an independent rate-one Poisson clock, and when the clock at v rings, $\sigma(v)$ is updated to agree with the majority of its four neighbors; in case of a tie, the new spin is chosen uniformly from $\{-1, +1\}$. Let $\mathbb{P}_p$ denote the law of the dynamics.

Define

$$p_c(T_4) := \inf\Big\{p \in [0,1] : \mathbb{P}_p\Big(\lim_{t\to\infty} \sigma_t(v) = +1 \text{ for every } v \in V\Big) = 1\Big\}.$$

Prove Morris' conjecture for T_4: for zero-temperature Glauber dynamics on the 4-regular tree,

$$p_c(T_4) = \frac{1}{2}.$$

Equivalently, for every $p > 1/2$, the process fixates to $+1$ at every vertex almost surely.

Table 3. Index of unsuccessful QED cases disclosed in the appendix.

Description	Area	Problem summary
Appendix B.1	Probability	Show that the invariant random proper edge-coloring measure on the d-regular tree is a factor of i.i.d.
Appendix B.2	Probability	Show that every finitely dependent $\mathrm{Aut}(T_d)$-invariant process on a finite state space is a factor of i.i.d.
Appendix B.3	Probability	Determine whether the three-dimensional self-avoiding walk has expected displacement larger than diffusive order, $d_3(n) \gg n^{1/2}$.
Appendix B.4	Probability	Prove Morris' conjecture $p_c(T_4) = 1/2$ for zero-temperature Glauber dynamics on the 4-regular tree.
Appendix B.5	Operator algebra	Determine whether coarse embeddability is a W^*-invariant for countable discrete groups.

B.5. Project 11

Problem statement. Is coarsely embeddability a W*-invariant? That is, if Γ, Λ are countable discrete groups and the group von Neumann algebras $L\Gamma \simeq L\Lambda$ are isomorphic, then Γ being coarsely embeddable into Hilbert spaces implies Λ is coarsely embeddable into Hilbert spaces as well.

C. Retry Information

In this section, A.1–A.5 refer to the 5 successful QED projects. The notation "P" refers to an internal problem statement within one of those appendices, not to the original research-project IDs from Section 5. For example, "A.4, P3" is the third internal problem statement within A.4, whereas Project 15 is the original research project corresponding to A.4.

Table 5. Simple/legacy-mode QED runs without decomposition, counted by verification rounds.

Run identifier	Verification rounds
A.4, P1	4
A.4, P2	4
A.5	1

Table 4. Per-attempt breakdown of proof generation runs in decomposition mode. **Attempt** = number of new plans generated. **Revisions** = same plan, number of times it is revised and the proof regenerated to completion. **Proof number** = same plan, number of times the proof is modified. Some successful appendix entries contain multiple internal problem statements, and not every internal problem statement necessarily appears in the final proof.

Run identifier	Attempt	Revisions	Proof number
A.1	2	1	2
A.2	1	2	5
A.3	1	1	1
A.4, P3	1	1	2
A.4, P4	1	1	2
A.4, P5	1	1	3
A.4, P6	1	1	2
A.4, P7	1	1	1
A.4, P8	1	3	5
A.4, P9	1	1	3
A.4, P10	1	1	1
A.4, P11	2	2	6
A.4, P12	2	3	3