LPBench: A Lean Benchmark for Research-Level Mathematical Theorem Proving*

Vaibhav Bajpai[1], Nathan Bowler[2], Johannes Carmesin[1], Jinghua Groppe[1], Sven Groppe[1], and Utku Okur[1]

[1]TU Freiberg
[2]Universität Hamburg

Abstract

In October 2026, OpenAI released hundreds of manuscripts resolving open mathematical problems, without publishing the full set of problems attempted by its model. Assessing the broader capabilities and limitations of such systems calls for independent benchmarks.

To this end, we introduce LPBench, a benchmark for research-level mathematics comprising 117 problems formalised in Lean 4, primarily in combinatorics and graph theory. As a first application, we evaluate Claude Opus 5.5 and Claude Fable 5.1, each with and without web search.

https://lp-bench.github.io/

*supported by the Krüger Stiftung and by Anthropic's AI for Science program.

1 Introduction

Artificial intelligence (AI) is producing a large number of mathematical results. Well-known examples include OpenAI's announced solution to the Navier–Stokes problem [18] and its release of more than 700 mathematical manuscripts [20]; a counterexample to the Jacobian conjecture in dimension three obtained using an Anthropic model [17, 26]; and AI-assisted resolutions of several of Erdős' problems [9, 10, 27, 28]. These are impressive achievements. However, a fuller assessment of the mathematical capabilities of AI systems also requires understanding which problems current models have failed to solve. For example, OpenAI reports that its manuscripts arose from an evaluation involving approximately 4000 problems [19], but the full set of attempted problems has not been made public.

A few months before OpenAI's announcement of the 700 manuscripts, some of us selected 10 challenges in Combinatorics together with formalisations in Lean[1], the solution of any one of which we would regard as a major result. Three of these ten problems were resolved in the OpenAI release: Hadwiger's conjecture, Sidorenko's conjecture and Ryser's hypergraph conjecture. Remarkably, all three were refuted by counterexamples. Although ten problems form a small sample, this suggests that current AI systems are more successful at disproving conjectures than at proving them.

To test such observations on a broader basis, we introduce LeanProofBench (LPBench), a formal and proof-based benchmark in Lean 4 for measuring the capabilities of AI systems in research-level mathematics. It consists of 117 problems, most of them in combinatorics and graph theory, and its first ten problems are those of Ten Challenges. The present paper describes the construction of LPBench and the protocol by which AI systems are evaluated on it. As a first application, we evaluate Claude Opus 5.5 and Claude Fable 5.1 on all 117 problems, each with and without web search.

The remainder of this manuscript is organised as follows. In Section 2, we select properties to compare benchmarks, and survey existing benchmarks. The next section, Section 3, introduces LPBench: the workshop in which the method took shape, how problems enter the benchmark, and the theorem sandwich for validating formal statements. Section 4 describes the composition of the problem set, and how open problems are posed. Section 5 describes how AI systems are evaluated on LPBench, covering the tools available to the agent, the session structure, the grading criteria and the quantities we record. Section 6 presents the evaluation data, and Section 7 analyses it. In Section 8, we compare LPBench with four public research-level formal benchmarks.

2 Benchmarking AI in Mathematics

As AI capabilities develop, mathematical benchmarks serve several purposes. They turn impressions of what AI systems can do into reproducible measurements that can be compared across systems. They identify weak spots of AI systems, and let mathematicians judge based on statistically sound evidence.

2.1 Properties to compare Benchmarks

To distinguish between benchmarks according to the problems they contain, we consider the following criteria:

(i) **Research-level mathematical depth.** The benchmark should contain problems whose difficulty and significance are recognised by active research mathematicians. The mere fact that a problem is unsolved does not guarantee that its solution constitutes a conceptual advance. Problems should therefore be selected by domain experts.

[1] https://tenchallenges.github.io/ten-challenges/

(ii) **Disciplinary breadth.** Mathematics encompasses a vast range of fields, many of which are under-represented in the current formalisation ecosystem. A benchmark covering diverse areas of mathematics is better positioned to assess genuine reasoning skills.

(iii) **Reproducible and objective evaluation.** A benchmark must establish unambiguous criteria for measuring mathematical performance, ensuring that results are reproducible and comparable across different AI systems and experimental settings.

(iv) **Independence.** The benchmark should be designed and governed by researchers with transparent funding and no commercial stake in the systems under evaluation. Furthermore, both benchmark maintenance and agent testing must be conducted on secure infrastructure.

2.2 Different Approaches in Benchmarking

To distinguish between benchmarks how they evaluate problems, we consider the following criteria:

(i) **Informal benchmark:** In an *informal* benchmark, the model writes its proofs in natural language, so they must be graded by human judges or by a large language model (LLM) acting as a judge.

(ii) **Formal benchmark:** A *formal* benchmark requires the AI system to produce output in a formal proof language so that the proof can be mechanically checked for correctness. Such a benchmark provides objective, scalable, and reproducible evaluation with little or no ongoing human grading cost, although they incur an overhead of formalisation work.

(iii) **Proof-based benchmark:** A *proof-based* benchmark requires the model to construct a full mathematical argument.

(iv) **Final-answer benchmark:** A *final-answer* benchmark only demands a specific piece of data (e.g. a number, a group, a graph) as the solution. As observed by Schmitt et al. [25], a correct final answer might not always reflect a full solution. The extent to which models exploit structural patterns or surface heuristics, rather than engage with the underlying reasoning, can be difficult to assess. (Proof-based benchmarks avoid this uncertainty, though at the cost of considerably greater evaluation complexity.)

2.3 The State of the Art: Existing Benchmarks

We survey benchmarks from the literature along the two axes above. Table 1 places them by their format.

Informal, final-answer	Informal, proof-based	Formal, proof-based
HLE	ProofBench (Ma et al.)	MiniF2F
FrontierMath	IMProofBench	PutnamBench
RealMath	FirstProof	CombiBench
LiveMathematicianBench	LemmaBench	FormalMATH
		ProofNet
		MSC-180
		AMBER
		FormalProofBench
		LeanEval
		ArXivLean
		Formal Conjectures
		LeanOpenProblems

Table 1: Existing benchmarks for mathematical capabilities, by format.

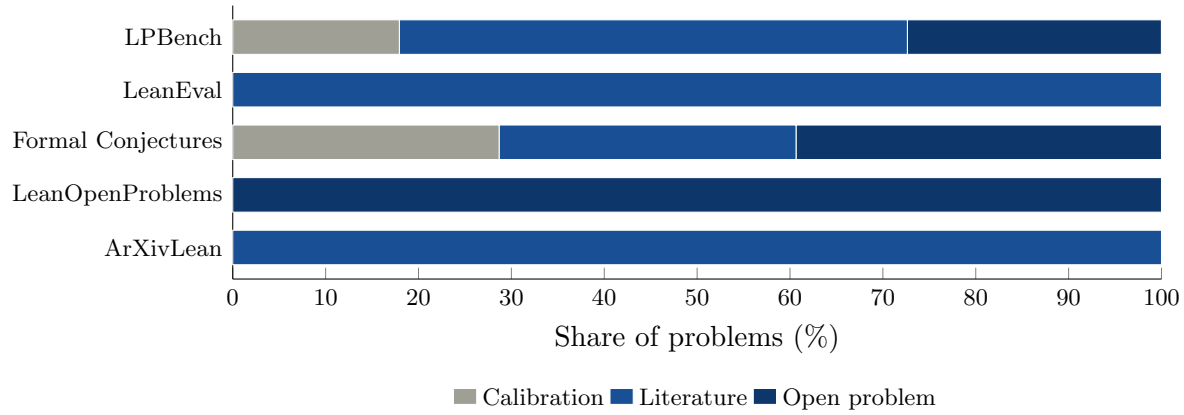

Figure 1: Distribution of five benchmarks across Calibration, Literature, and Open problem groups. Six formal benchmarks of calibration level are omitted. For other formal benchmarks considered in this paper, the data is not publicly available.

Informal, proof-based benchmarks

(i) **IMProofBench**[2] [25]. A research-level benchmark consisting of 39 peer-reviewed problems developed by expert mathematicians. Each problem requires a detailed proof, which is graded by the problem's author. The evaluation simulates a research environment with tools such as web search and SageMath. In this agentic setting, the paper reported fully correct proofs for 22% of the problems with GPT-5 and 19% with Grok 4. The live leaderboard now lists GPT-5.4 at 49.3% and Gemini 3.1 Pro Preview at 45.7%.[3]

(ii) **FirstProof** [1]. An experimental prototype consisting of ten original, unpublished research-level mathematics questions drawn directly from the active research of the authors (in algebraic combinatorics, spectral graph theory, algebraic topology, and other fields). The answers were initially encrypted and released after a fixed date. Both GPT-5.2 Pro and Gemini 3.0 Deep Think struggled to answer many of the questions.

(iii) **LemmaBench** [22]. A live benchmark of self-contained lemmata extracted from recent arXiv preprints. Proofs are evaluated both by an LLM judge and by human experts.

(iv) **ProofBench (Ma et al.)** [16]. An expert-annotated dataset of 435 natural-language proofs, written by o3, Gemini 2.5 Pro and DeepSeek-R1 for 145 problems from six mathematics competitions including the IMO, USAMO and Putnam, each graded by experts on a 0–7 scale. Its purpose is to evaluate automatic proof graders rather than provers. The authors use it to develop ProofGrader, an LLM-based evaluator whose scores differ from the experts' by less than one point on average.

Informal, final-answer benchmarks

(i) **LiveMathematicianBench** [13]. A dynamic multiple-choice benchmark for research-level mathematical reasoning built from recent arXiv papers published after model training cutoffs. It introduces a thirteen-category logical taxonomy of theorem types to generate plausible

[2] https://improofbench.math.ethz.ch/
[3] Accessed 19 October 2026.

alternative statements which are either false or too weak. Even the best-performing model (Gemini 3.1 Pro Preview) achieves only 43.5% accuracy.

(ii) **FrontierMath** [12]. A benchmark of hundreds of original, exceptionally challenging mathematics problems crafted by expert mathematicians across most major branches of modern mathematics. The answer format is a Python object, typically a natural number. FrontierMath uses new, unpublished problems and automated verification to minimise data contamination. At its release, FrontierMath posed a severe challenge to frontier models, with very low reported solve rates. A 2026 audit addressed errors in 42% of the problems. Tier 4, the hardest level, opened in July 2025 with a top score of 5%; within 14 months the top score reached 98%, and Epoch AI now considers it saturated. It has since turned to *FrontierMath: Open Problems*[4], a collection of about 50 unsolved research problems, of which its page lists five as solved by AI and five as solved by humans and AI together.

(iii) **HLE (Humanity's Last Exam)** [23]. A broad, multidisciplinary benchmark containing thousands of expert-level problems, not exclusively mathematical. It consists of multiple-choice and short-answer questions designed to resist memorisation and internet lookup. At launch, the best models achieved low accuracy, though performance on HLE has risen substantially as frontier models have improved.

(iv) **RealMath** [33]. A continuous benchmark derived directly from research papers (arXiv) and mathematical forums (Stack Exchange), designed to mitigate data contamination by using newly published papers. Evaluation focuses on verifiable answers rather than proof assessment, allowing for automated correctness checking.

Formal, proof-based benchmarks

(i) **MiniF2F** [34]. A cross-system benchmark for formal Olympiad-level mathematics. It contains 488 competition problems (from AMC, AIME and IMO), split evenly into validation and test sets and formalised in Lean and other proof assistants. MiniF2F has driven substantial progress in neural theorem proving. Seed-Prover [7] now proves 243 of the 244 test problems (99.6%). A later review [21] found that the formal and informal statements disagree in more than half of the problems, an instance of the mismatch risk discussed in Section 3.3.

(ii) **PutnamBench** [30]. A benchmark of 672 problems from the Putnam competition formalised in Lean 4, with further formalisations in Isabelle and Rocq. Its Lean track is complete, since proofs of all 672 problems have been reported with Claude Opus 5, and Aleph Prover solves 668 of them.[5]

(iii) **FormalMATH** [32]. A large-scale Lean 4 benchmark comprising 5,560 formally verified problems spanning high-school Olympiad challenges to undergraduate-level theorems across algebra, geometry, calculus, number theory, and discrete mathematics. At publication the strongest model achieved a 16.46% success rate. A 2026 audit certified 141 of its statements as defective [4].

(iv) **MSC-180** [14]. A benchmark organised according to the Mathematics Subject Classification (MSC2020), comprising 180 formal verification problems in Lean 4 (3 from each of 60 mathematical branches), spanning undergraduate to graduate levels. The best model achieves only an 18.89% overall pass rate.

[4] https://epoch.ai/frontiermath/open-problems, accessed 30 October 2026.
[5] https://trishullab.github.io/PutnamBench/leaderboard.html, accessed 19 October 2026.

(v) **ProofNet** [5]. A benchmark of 371 problems from undergraduate analysis, algebra and topology, formalised in Lean 3.

(vi) **CombiBench** [15]. The first comprehensive benchmark specifically focused on combinatorial problems formalised in Lean 4. It comprises 100 problems ranging from middle-school level to IMO and undergraduate level, covering over ten combinatorial topics. CombiBench also introduces Fine-Eval, a standardised evaluation framework for fill-in-the-blank problems. All tested models struggle significantly, with the best (Kimina-Prover) solving only 7 out of 100 problems.

(vii) **AMBER** [31]. A benchmark for applied mathematics in Lean 4, covering convex analysis, optimisation, numerical algebra, and high-dimensional probability. AMBER enforces a "construction-verification" workflow where the agent must synthesize explicit solutions before proving their correctness. Problems range from standard undergraduate exercises to PhD-level research problems.

(viii) **FormalProofBench** [24]. A benchmark from Vals AI of advanced undergraduate and graduate-level problems, with a public and a private split of 100 problems each. Each task pairs a natural-language problem with a Lean 4 formal statement, and the agent has up to 40 turns with a Lean checker and a Mathlib search tool [29]. The March 2026 paper reported 33.5% for the best model. By October 2026 the live version, ProofBench (Vals AI), stands at 100% for Claude Fable 5.1, Claude Opus 5.5, Claude Sonnet 5.5 and AlephProver, and at 99% for GPT-6 Astra, GPT-6.1 Sol, Gemini 4 Argon and Claude Opus 5.[6]

(ix) **LeanEval**[7]. A benchmark for formal mathematics in Lean 4, maintained by the Lean FRO and graded by Comparator. Benchmark authors write trusted problem statements in shared Lean modules, and the tooling generates isolated Comparator workspaces for each problem. A submission is scored entirely by Comparator, so a problem counts as solved if and only if Comparator accepts the submitted solution, which rules out benchmark-gaming via definition alteration or axiom introduction. The benchmark features hard formalisation problems, usually with a known informal solution. Many of them were prepared with the help of AI tools (Aristotle, Claude Code and Codex) and reviewed before acceptance. Public results from AI provers are reported on a live leaderboard.

(x) **ArXivLean**[8] [8]. A research-level benchmark from ETH Zurich and INSAIT, updated every three months with problems extracted from recent arXiv papers and formalised in Lean with the help of AI, followed by manual review. On its first release, the best models proved about 17% of the problems.

(xi) **Formal Conjectures**[9] [11]. An open-source collection of research-level problems formalised in Lean 4, hosted by Google DeepMind and maintained by a community of contributors. Its first preprint (May 2026) lists 2,615 statements: 1,029 open conjectures, intended as a contamination-free test of proof discovery, and 836 solved problems, intended for autoformalisation. The problems come from sources such as the Erdős problems, Wikipedia, MathOverflow, the OEIS and arXiv papers. The correctness of the formalisations rests on community review, with proofs and disproofs found by AI systems serving as an additional audit.

[6] https://www.vals.ai/benchmarks/proof_bench, accessed 30 October 2026.
[7] https://lean-lang.org/eval/
[8] https://matharena.ai/arxivlean/
[9] https://github.com/google-deepmind/formal-conjectures

(xii) **LeanOpenProblems**[10] (Epoch AI). An open-source evaluation framework built on Inspect AI, in which an agent is given a conjecture stated in Lean 4 and asked to prove or disprove it. Two benchmarks are built on it. *OEIS Open* [2] consists of 492 open conjectures from the On-Line Encyclopedia of Integer Sequences, formalised in Lean. With a budget of $50 per attempt, language models resolve 147 of them (30%), and with $200 per attempt the best current model scores 44% on a subset of 100. *FrontierMath Erdős* [3] consists of 68 Erdős problems open as of August 2026, selected by Thomas Bloom for significance and difficulty. Of the statements, 50 come from Google DeepMind's Formal Conjectures collection, and the remaining 18 were formalised with the help of AI and are still under expert review. With one attempt per problem, a budget of $300 and 72 hours, the best model (GPT-6 Astra) resolves 2 of the 68 problems (3%), and the other four models evaluated resolve none.

The survey shows that *(i)* competition-level formal theorem proving has reached saturation, *(ii)* undergraduate and graduate-level formal benchmarks are following, with the Lean track of PutnamBench and FormalProofBench solved at 100% for several frontier models as of October 2026, and *(iii)* formal benchmarks at the research level are now appearing, including LeanEval, ArXivLean, as well as OEIS Open and FrontierMath Erdős, where the latter two are built on LeanOpenProblems.

3 The ProofBench Initiative

LPBench is a Lean-based benchmark currently comprising 117 problems concentrated in combinatorics and graph theory. It poses open conjectures with both prove and disprove options, alongside published research results. And it fixes a complete evaluation protocol, from the agent's tools and budgets to the measurements taken for every attempt, such as tokens, time, cost and proof length (Section 5), so results obtained by different groups can be compared. The procurement method described in Section 3.2 accepts problems from any field, and we expect the scope to widen as other groups contribute. The project was initially called ProofBench, and the workshop in which it began took place under that name.

3.1 The ProofBench Workshop

The first steps of the project were taken at the ProofBench workshop at Technische Universität Bergakademie Freiberg from 22 September to 3 October 2025. Approximately twenty participants took part, including research mathematicians with a combinatorics background and a smaller group of Lean experts. The objectives of the workshop were *(i)* to lay down the standards that a Lean-based benchmark of research-level mathematics should meet, *(ii)* to formalise a current research result in Lean 4 to serve as a test problem for existing AI systems, and *(iii)* to bring together experts in pure mathematics, interactive theorem proving and machine learning.

At the workshop, as a prototype for the 117 problems in our project, two problems were formalised: that every finite graph is ubiquitous for the minor relation and the existence of a counterexample to Andreae's ubiquity conjecture [6]. At the time AI tools did not prove very useful and everything was done manually. For the formalisation, the participants worked out the *blueprint*: a document that maps the implications between the statements that contribute to the proof of the main theorem, and then specified the definitions needed, such as vertex connectivity, minors, making design choices along the way. This required navigating significant gaps in Mathlib [29], Lean's mathematical library, and resolving subtle discrepancies between natural-language and formal statements. The definitions became the first modules of the statement library of LPBench and are reused by other problems. The idea of the theorem sandwich (q.v. Section 3.3) also emerged from the formalisation process of the workshop.

[10] https://github.com/epoch-research/LeanOpenProblems

3.2 How Problems Enter the Benchmark

We use the following method for the procurement of problems.

(i) A researcher or research group proposes a result, a conjecture, or a question from their field.

(ii) Our team formalises the statement in Lean 4, either together with the researcher or independently, using AI tools under human review. During our project, we observed that the tools for auto-formalisation into Lean have advanced remarkably within the last year.

(iii) The statement and the definitions it needs enter the benchmark. Any reference proof or other related material go to the proctor library described in Section 4.

3.3 Quality Control via the Theorem Sandwich

Formalising a theorem statement carries a significant risk that the resulting Lean 4 code does not match the author's intended meaning. The MiniF2F review cited in Section 2.3 shows how common such mismatches are, and a 2026 audit of five Lean benchmarks found 398 statements with a machine-checkable defect, such as being unprovable or vacuous [4].

We guard against this by a routine where we relate the statement in question to its neighbours. We call the resulting implications a *theorem sandwich*: alongside the statement we formalise an *upper bound*, a stronger statement, typically a set of key lemmata or a known or conjectured stronger result, and a *lower bound*, a handful of derived corollaries. A statement that is accidentally too strong makes the implication from the upper bound unprovable, and one that is accidentally too weak fails to yield the corollaries. The lemmata of the upper bound may themselves be left unproved, marked `sorry`, Lean's placeholder for a missing proof. For an open problem this is unavoidable, since proving all of them would end up solving the problem itself.

4 The Problem Set

LPBench currently contains 117 problems, the majority of them in graph theory.

Definitions that are needed to *state* problems but are missing from Mathlib live in a statement library, which the evaluated systems can see and import. Everything that would help to *solve* problems—the theorem sandwiches, reference proofs and auxiliary results—lives in a separate proctor library that never enters the evaluation environment. The problem files and the statement library are public[11].

A database accompanies the problems. For every problem it records the name, the category, the MSC2020 class, the formal statement with a plain-language summary, the literature references, and the formal and informal statements of the bounds of its theorem sandwich; for some problems it also contains a complete informal proof in LaTeX. A version of the database restricted to the statements is public together with the problem files.

Nineteen problems have complete reference solutions. They allow us to test the grading pipeline end to end and give an upper bound on the length of the shortest formal proof. For each of the 32 open problems we also provide its negation as a second Lean statement. A system answers an open problem by proving either of the two.

[11] https://github.com/lp-bench/lp-bench

Category	Problems	Description
Open	32	open conjectures, posed in prove-or-disprove form
Literature	64	literature results with a published proof
Calibration	21	elementary exercises
Total	117	

Table 2: Composition of LPBench.

5 Evaluation Process

5.1 Harness

The *harness* is the software that runs an agent. It supplies the tools, enforces the budgets and records the results. Let us use *model* to mean the neural network and *agent* to refer to the model together with the harness. We use Inspect AI[12], the open-source framework of the UK AI Security Institute, which is also used by IMProofBench and LeanOpenProblems. Commercial models are accessed through their providers' application programming interfaces (APIs). Open-weight models are accessed through the same OpenAI-compatible interface, so they can be served by an academic inference service or self-hosted on a high-performance computing cluster. We do not use ready-made tools such as Claude Code or Codex, because each ships its own harness, which differs between vendors and is updated without notice. This makes results hard to compare or reproduce, so we run every model in the same harness instead. The price is that a model trained to work with its vendor's own harness may perform somewhat worse in ours.

The harness is open source[13], and all results in this paper were produced with it. We also provide the QEDRank website[14], an independent tool with an evaluation pipeline of its own, which evaluates models on a corpus of formal problems and compares Lean benchmarks with each other.

Sandbox. Each problem is attempted in its own container, built from an image that contains only the Lean toolchain, Mathlib and the statement library of the problems in the run. The container has no network access, except in runs with web search (see below), so the agent cannot download papers or packages, and it has no access to the host's environment variables and in particular no API keys, nor to other problems or reference solutions. The containers run under Apptainer, which needs no administrator rights, through our open-source sandbox package `inspect-apptainer`, which ships with the harness as the folder `inspectAI-apptainer` of its repository[15].

The agent has the following tools.

- A shell inside the sandbox, where the Lean toolchain, Mathlib and the statement library are installed, so that the agent can compile at any time.
- The Lean language server through `lean-lsp-mcp`, providing goal states, diagnostics and local Mathlib search. Its five web-backed search tools (Loogle, LeanSearch, LeanFinder, state search and premise search) are available only in runs with network access.
- In a separate configuration, web search through one external provider for all models rather than each vendor's built-in search, so that every model sees the same search engine.
- A file-editing tool, a Python interpreter with SymPy and NetworkX, SageMath (which includes GAP and nauty), and the Boolean satisfiability (SAT) solver CaDiCaL.

[12] https://inspect.aisi.org.uk/
[13] https://github.com/lp-bench/QEDRank
[14] https://qedrank.vercel.app/, accessed 30 October 2026.
[15] https://github.com/lp-bench/QEDRank/tree/main/inspectAI-apptainer

5.2 Sessions, Limits and Handoffs

Each model attempts each problem in a fixed number of sessions, four in this paper. Each session starts with a "fresh context", in other words, the initial text the model reads before each reply is empty and nothing carries over from earlier sessions except what the harness supplies. In each session, the agent only sees the current state of the files, the Lean compiler output for them, and a handoff note written in the previous session in a fixed format listing proved lemmata, open goals, approaches that failed, and any Lean code drafted but not yet saved. After every session the first grading layer of Section 5.4 is applied. The first session that passes ends the chain, and the full grading follows.

We chose fresh sessions with a handoff over two alternatives. Continuing a single conversation across all sessions has three drawbacks. The entire history is present at every turn, so later sessions cost far more tokens than early ones. Long transcripts of stale compiler output degrade reasoning. Finally, context windows differ between models and automatic context compaction, in which the model summarises its own history once its context limit is reached, depends on the size of the context window and on how well the model summarises, and would thus make the comparison depend on an uncontrolled component. The fixed handoff format keeps sessions comparable across models, at the price of detail lost when a note is poor.

Each session carries an output-token budget enforced by the harness: a session ends after 120,000 output tokens, reasoning tokens included, and the handoff note has a separate budget of 32,000. A single reply is capped at 64,000 output tokens and a single tool call at 30 minutes. As safeguards, each session is also capped at 40 million total tokens and each attempt at 18 hours. These limits are the harness defaults and were used unchanged in the runs of this paper.

Cached input is prompt content that the provider reuses from earlier model calls and bills at a reduced rate. We record the percentage of input tokens served from the provider's prompt cache and calculate costs at the provider's list prices, accounting for the applicable cache pricing.

5.3 Secure Testing

An interesting question naturally arises when an AI system is being evaluated.

Question. *To what extent does the prior publication of proofs, and of the problem statements themselves, affect the integrity of the evaluation? Does a benchmark whose problems are public still measure reasoning rather than recollection from training data?*

Since the problem statements are public (Section 4), they may enter the training data of future AI models. For the literature problems, informal proofs are published already. We mitigate the risk that a model has already seen the solution in three ways. We evaluate an AI model twice, with and without access to external search. We record the release date of every model, so a model released before the publication of the problem set cannot have trained on the statements. And we will monitor the relative progress of systems on problems whose proof is publicly available and on those where it is not.

5.4 Grading

A submission is graded in two layers, following the designs of LeanEval and LeanOpenProblems, in a fresh container that receives only the source text of the submitted files.

Layer 1: static check and compilation. The theorem statement and every definition of the problem file must be unchanged. Imports are restricted to Mathlib and its archive, the LPBench statement library, the matroid library of Peter Nelson[16] and the agent's own helper files. The files

[16] https://github.com/apnelson1/Matroid

must compile, and the main theorem may depend only on Lean's standard axioms.

Layer 2: Comparator. If the first layer passes, the submission is checked a second time by Comparator[17], the proof checker that the Lean FRO built for adversarial settings. The check runs in an isolated environment without administrator rights, so the submitted code cannot interfere with it.

A submission counts as a solution only if both layers accept it. Accepted solutions are then audited by hand by the maintainers, except for calibration problems, whose solutions are audited only when they are shorter than the reference proof or when no reference proof exists yet.

5.5 Measurements

For each attempt, we record: **outcome** (pass/fail and number of sessions used); **model and tool usage** (input and output token counts, percentage of cached input tokens, numbers of model and tool calls, and maximum prompt length in tokens); **cost and time** (cost at provider list prices, total time including grading, and working time); and **proof statistics** (proof length and compilation time). Working time excludes time spent waiting for an available provider connection or for a retry after a failed request.

Per run, we freeze and record the harness commit, the sandbox image, the Lean toolchain and Mathlib commit, the LPBench commit of the problem set, the model identifier, its reasoning-effort setting, and whether web search was enabled.

6 Evaluation Data

The runs share one configuration, recorded in each run's manifest: evaluation code hash `9d6e862b64f7` and harness tags official-v1, v1.1 and v1.2 (commits 9e812aa, 9c24ac3 and c5ab276, which leave the evaluation code unchanged), sandbox image `b41c11aaf9ec37f2`, Lean 4.34.1, Mathlib commit `d13f23b`, ProofBench commit `5950904`, and the session limits of Section 5.2. Reasoning effort: xhigh. Each configuration (a model, with or without web search) attempted every problem exactly once. The 117 problems of a configuration were run in three batches of 39, each as a separate cluster job. The run logs, the grade records and the mounted cache are archived under Zenodo concept 23857356. The tables and figures were generated from them.

Four configurations were run: Claude Opus 5.5 and Claude Fable 5.1, each with and without web search, one attempt per problem of up to four sessions. The results are summarised in Table 3, and the measured quantities of every attempt are in Section A, one table per configuration.

Model	Web search	Total	Calibration	Literature	Open	Errors found	Cost (USD)
Claude Fable 5.1	no	42/117	20/21	22/64	0/32	0/1	1,413.72
Claude Fable 5.1	yes	51/117	21/21	30/64	0/32	1/1	1,468.25
Claude Opus 5.5	no	45/117	21/21	24/64	0/32	1/1	391.23
Claude Opus 5.5	yes	55/117	21/21	34/64	0/32	1/1	478.68

Table 3: Problems solved out of problems graded, one attempt per problem with up to four sessions. Errors found: flawed statements the model objected to, out of the statements found flawed after the runs; see Section 7.4 for more information. Cost at the providers' list prices.

7 Analysis of the Results

7.1 Comparison of Models

Comparisons between configurations are paired, since all attempt the same problems; we count the problems solved by exactly one of the two and apply McNemar's exact test. The two models are

[17] https://github.com/leanprover/comparator

indistinguishable at this sample size: without search Fable solved 2 problems that Opus did not and Opus 5 that Fable did not ($p = 0.45$); with search, 2 against 6 ($p = 0.29$). Three observations put the small difference in Opus's favour in context. First, Fable 5.1 was released on 1 September 2026 and Opus 5.5 on 22 September 2026, and Anthropic introduced Opus 5.5 as reaching Fable 5.1's level on most tasks at lower cost[18]. Second, the budgets were identical, four sessions of 120,000 output tokens per problem, so resources do not explain the difference. Fable used more of its budget than Opus on the same problems, and most of that additional output went into problems it did not solve. At these budgets, its extra effort did not turn into extra solutions. Third, Anthropic does not publish the sizes of its models, so the price difference is only an indirect indication of their relative sizes, and nothing concrete about capability can be inferred from it.

7.2 Web search and contamination

Web search clearly helps each model solve more problems. Fable solved 10 problems only with search and 1 only without ($p = 0.012$), Opus 10 and 0 ($p = 0.002$). Seven of the ten are the same problems for both models. For four of them (P004, P005, P015, P054) a complete public Lean proof exists on GitHub. Fetched Lean code also entered the solutions of P011, P027, P052 and P056. Search was used in 72 and 68 of the 117 attempts of the two search runs, yet most solves used none: 35 of Fable's 51 and 41 of Opus's 55.

7.3 Sessions and effort

Figures 2 and 3 show the problems newly solved in each session and the cost per attempt by category, respectively. Most problems were solved in the first session. The models stop on their own: of the 1,337 sessions, 135 (10%) ended at the budget of 120,000 output tokens and the rest with the model submitting. Ten attempts ran for more than 8 hours, all unsolved. In them the model ran computations of its own, SAT solvers and enumeration programs, in tool calls of up to 30 minutes each.

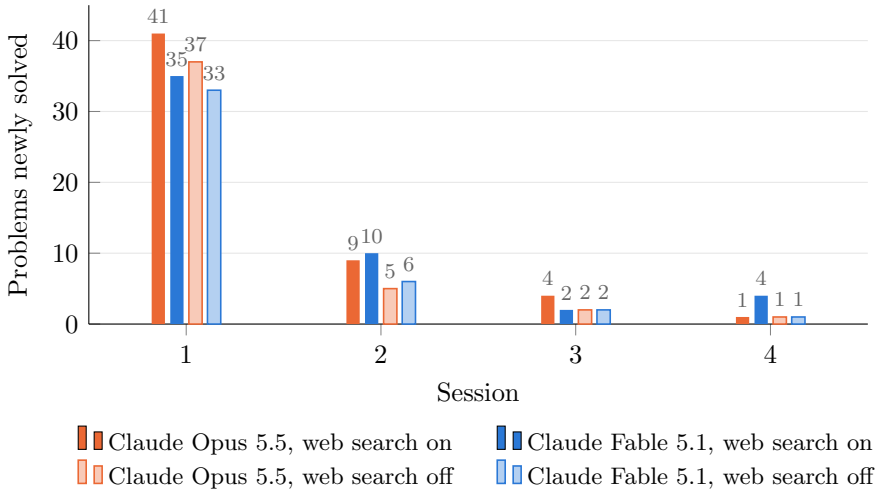

Figure 2: Problems newly solved in each session, one attempt per problem with up to four sessions. A later session starts from the files and the handoff note of the previous one.

[18] Anthropic, *Introducing Claude Opus 5.5*, 22 September 2026, https://www.anthropic.com/claude-opus-5-5

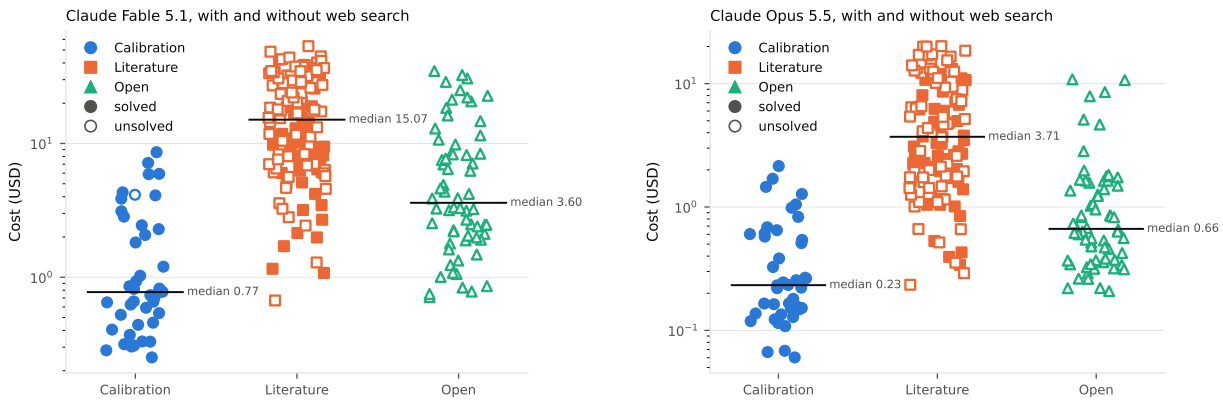

Figure 3: Cost per attempt by category, log scale; each panel pools the attempts of a model with and without web search, two per problem. Filled markers: solved; hollow: unsolved; bar: median of the category. Fable's medians are three to five times those of Opus, and for both models the unsolved attempts are the expensive ones.

7.4 Objections by Test-takers

For a problem that has no negated statement, an agent convinced that the formal statement is false may, instead of a proof, submit a text beginning with "FALSE" and a short explanation. This objection ends the attempt, which counts as unsolved, and is reviewed by hand. Open problems do not need this mechanism. For an open problem, both the statement and its negation are offered, and a disproof is graded like a proof.

Five objections were raised. Three concerned P099, independently by Fable with search and by Opus with and without search, and they were valid: the formal statement left out a subtle assumption that is standard in its field and therefore implicit in the source. The other two, by Fable without search against P001 and P004, were not. P099 was corrected in the next commit of the problem repository, 84b49e2, under the same number. A later commit, 37807a7, reclassified P005, the Erdős–Hajnal conjecture for paths, which was open when it was added: in October 2026 Nguyen announced a proof (arXiv:2610.24887), and Google DeepMind released a Lean formalization of a proof found by its Cogentic system. Both models solved P005 only with web search, by importing that development, and without search neither did. We count P005 as a literature problem, which is what it was at the time of the runs, and all counts in this paper include it there.

8 Comparison with Other Benchmarks

We compare LPBench with four of the research-level formal benchmarks surveyed above. Section 8.1 shows which subjects the problems cover, and Section 8.2 measures how long the statements are.

8.1 Subject coverage

Problems are counted by the subject labels their benchmark provides, mapped onto the eight areas of Table 4, which gives the share of labelled problems in each.

Benchmark	Logic	Combinatorics	Number theory	Algebra	Analysis	Geometry/topology	Probability	Applied	Other	n
LPBench	1	87	4	3	0	4	1	0	0	117
LeanEval	2	8	28	16	13	32	0	1	1	128
Formal Conjectures	1	29	49	6	5	7	1	1	0	4885
LeanOpenProblems	0	27	55	4	4	9	0	1	0	244
ArXivLean	4	51	7	5	8	5	8	4	8	53

Table 4: Subject coverage: share of labelled problems (%) by area; a row sums to 100 up to rounding. ArXivLean-0926: primary subject of each of its 53 problems, counted by hand from the Hugging Face dataset. The other rows are measured from the benchmarks' own labels.

8.2 Statement complexity

We measure the length of a statement in Lean tokens, the units Lean reads the source as: identifiers, numbers, literals and operator symbols, each counted once. The count covers the theorem's hypotheses and conclusion together with every definition they use from the benchmark's own files, followed transitively; Mathlib is excluded. The same rule is applied to every benchmark. As Figure 4 shows, the statements of LPBench are the longest of the five benchmarks.

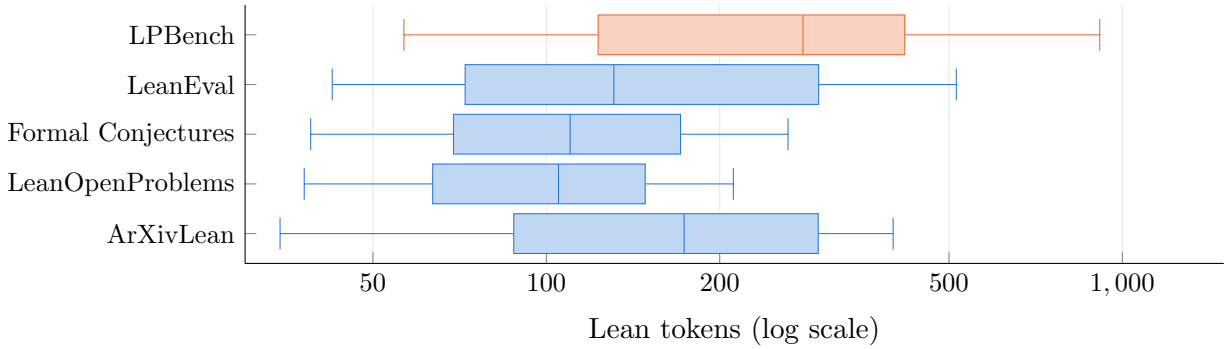

Figure 4: Statement length by benchmark, in Lean tokens. Box: middle half of the problems; bar: median; whiskers: 10th to 90th percentile; log scale.

Acknowledgements

The authors gratefully acknowledge the computational resources provided by the high performance computing cluster at TU Freiberg.

References

[1] Mohammed Abouzaid et al. *First Proof.* 2026. arXiv: 2602.05192 [cs.AI].

[2] Tom Adamczewski. *OEIS Open: How Many Conjectures Can Language Models Turn into Theorems?* 2026. arXiv: 2608.11941 [cs.AI].

[3] Tom Adamczewski and Thomas F. Bloom. *FrontierMath Erdős*. 2026. arXiv: 2609.25050 [cs.CL].

[4] Pawan Sasanka Ammanamanchi, Siddharth Bhat, and Stella Biderman. *Faults in Our Formal Benchmarking: Dataset Defects and Evaluation Failures in Lean Theorem Proving*. ICML 2026. 2026. arXiv: 2606.29493 [cs.AI].

[5] Zhangir Azerbayev et al. *ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics*. 2023. arXiv: 2302.12433 [cs.CL].

[6] Johannes Carmesin. "On Andreae's ubiquity conjecture". In: *Journal of Combinatorial Theory, Series B* 162 (2023), pp. 68–70.

[7] Luoxin Chen et al. *Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving*. 2025. arXiv: 2507.23726 [cs.AI].

[8] Jasper Dekoninck et al. *Beyond Benchmarks: MathArena as an Evaluation Platform for Mathematics with LLMs*. 2026. arXiv: 2605.00674.

[9] *Erdős Problem #1196*. 2026. URL: https://www.erdosproblems.com/1196 (visited on 2026-04-29).

[10] Tony Feng et al. *Semi-Autonomous Mathematics Discovery with Gemini: A Case Study on the Erdős Problems*. 2026. arXiv: 2601.22401 [cs.AI].

[11] Moritz Firsching et al. *Formal Conjectures: An Open and Evolving Benchmark for Verified Discovery in Mathematics*. 2026. arXiv: 2605.13171.

[12] Elliot Glazer et al. *FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI*. 2025. arXiv: 2411.04872 [cs.AI].

[13] Linyang He, Hanze Dong, et al. *LiveMathematicianBench: A Live Benchmark for Mathematician-Level Reasoning with Proof Sketches*. 2026. arXiv: 2604.01754 [cs.CL].

[14] Sirui Li et al. *MSC-180: A Benchmark for Automated Formal Theorem Proving from Mathematical Subject Classification*. 2025. arXiv: 2512.18256 [cs.AI].

[15] Junqi Liu et al. *CombiBench: Benchmarking LLM Capability for Combinatorial Mathematics*. 2025. arXiv: 2505.03171 [cs.AI].

[16] Wenjie Ma et al. *Reliable Fine-Grained Evaluation of Natural Language Math Proofs*. 2025. arXiv: 2510.13888 [cs.CL].

[17] Guowu Meng and Liang Yang. *A five-variable counterexample to the Hessian conjecture, and the low-dimensional status of the Jacobian and Hessian conjectures*. 2026. arXiv: 2607.22198.

[18] OpenAI. *Finite Time Blowup for Navier–Stokes*. Sept. 2026. URL: https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf.

[19] OpenAI. *OpenAI math repository: README*. Oct. 2026. URL: https://github.com/openai/math/blob/4eaee87548250b2682affda7d05d947c05554bae/README.md.

[20] OpenAI. *OpenAI Research Catalog: 373 result families in 721 manuscripts*. Oct. 2026. URL: https://github.com/openai/math/blob/main/overview.pdf.

[21] Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. "miniF2F-Lean Revisited: Reviewing Limitations and Charting a Path Forward". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2025. URL: https://openreview.net/forum?id=KtaHv0YUyh.

[22] Antoine Peyronnet et al. *LemmaBench: A Live, Research-Level Benchmark to Evaluate LLM Capabilities in Mathematics*. 2026. arXiv: 2602.24173 [cs.AI].

[23] Long Phan et al. “A benchmark of expert-level academic questions to assess AI capabilities”. In: *Nature* 649.8099 (Jan. 2026), pp. 1139–1146. ISSN: 1476-4687. DOI: 10.1038/s41586-025-09962-4.

[24] Nikil Ravi et al. *FormalProofBench: Can Models Write Graduate Level Math Proofs That Are Formally Verified?* 2026. arXiv: 2603.26996 [cs.AI].

[25] Johannes Schmitt et al. *IMProofBench: Benchmarking AI on Research-Level Mathematical Proof Generation*. 2025. arXiv: 2509.26076 [cs.CL].

[26] ScienceDaily. *Claude Fable 5 AI finds a tiny formula that topples an 87-year-old math conjecture*. Aug. 2026. URL: https://www.sciencedaily.com/releases/2026/08/260804034634.htm.

[27] Nat Sothanaphan. *Resolution of Erdős Problem #728: a writeup of Aristotle's Lean proof*. 2026. arXiv: 2601.07421 [math.NT].

[28] Terence Tao. *The story of Erdős problem #1026*. What's new. Dec. 2025. URL: https://terrytao.wordpress.com/2025/12/08/the-story-of-erdos-problem-126/ (visited on 2026-05-10).

[29] The mathlib Community. “The Lean Mathematical Library”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*. ACM, 2020, pp. 367–381. DOI: 10.1145/3372885.3373824.

[30] George Tsoukalas et al. *PutnamBench: Evaluating Neural Theorem-Provers on the Putnam Mathematical Competition*. 2024. arXiv: 2407.11214 [cs.AI].

[31] Bowen Yang et al. *Construction-Verification: A Benchmark for Applied Mathematics in Lean 4*. 2026. arXiv: 2602.01291 [cs.LO].

[32] Zhouliang Yu et al. *FormalMATH: Benchmarking Formal Mathematical Reasoning of Large Language Models*. 2025. arXiv: 2505.02735 [cs.AI].

[33] Jie Zhang et al. *RealMath: A Continuous Benchmark for Evaluating Language Models on Research-Level Mathematics*. 2025. arXiv: 2505.12575 [cs.AI].

[34] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. *MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics*. 2022. arXiv: 2109.00110 [cs.AI].

A Measured Quantities of Every Attempt

Table 5: Measured quantities of Claude Fable 5.1 where web search is not allowed.

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
Calibration (20/21 solved)													
P013	✓	1	2.12	95	68	5.90	13.5	13.5	37	31	94	331	9

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
P014	✓	1	0.34	94	9	0.77	2.8	2.7	17	17	27	131	9
P022	✓	1	0.74	93	33	2.44	10.6	7.5	18	19	58	154	7
P029	✓	1	0.20	90	11	0.82	3.0	3.0	10	10	26	96	10
P034	✓	1	0.16	89	8	0.65	2.4	2.4	9	9	23	53	10
P035	✓	1	0.11	87	8	0.59	2.3	2.3	7	7	20	57	10
P036	✓	1	0.15	88	10	0.78	2.8	2.8	9	9	25	92	10
P041	✓	1	0.08	88	4	0.31	1.3	1.3	6	6	15	144	6
P083	×	4	1.84	95	50	4.14	11.8	11.7	43	46	76	–	–
P084	✓	1	3.84	97	103	8.61	24.7	24.7	40	40	132	665	24
P087	✓	1	2.76	96	87	7.16	19.6	19.5	39	39	110	660	26
P088	✓	1	0.27	94	12	0.86	3.1	3.1	19	15	23	92	9
P089	✓	1	0.15	89	8	0.66	2.7	2.7	9	9	24	151	13
P090	✓	1	0.18	90	9	0.69	2.6	2.6	10	10	25	36	9
P091	✓	1	0.46	92	30	2.07	8.5	8.5	14	14	44	166	38
P092	✓	1	0.12	87	4	0.41	2.1	2.1	7	7	21	363	15
P093	✓	1	1.05	93	40	3.11	9.0	9.0	22	22	77	231	14
P098	✓	1	0.12	88	5	0.46	3.1	3.0	8	8	21	29	19
P109	✓	1	0.11	91	6	0.44	2.7	2.7	10	8	16	56	14
P110	✓	1	0.07	90	4	0.28	1.6	1.6	6	6	13	25	9
P111	✓	1	0.11	93	4	0.30	1.6	1.6	12	10	14	29	7
Literature (22/64 solved)													
P001	×	1	0.06	75	9	0.67	2.2	2.2	7	4	22	–	–
P002	×	4	0.54	85	49	3.58	10.5	10.5	22	26	31	–	–
P004	×	4	0.36	81	47	3.25	10.9	10.9	18	21	26	–	–
P005	×	4	0.24	76	41	2.81	9.1	9.1	10	14	24	–	–
P008	×	4	4.11	95	211	15.48	150.6	150.6	50	54	132	–	–
P011	✓	1	3.57	95	107	9.42	22.8	22.8	36	36	151	685	27
P012	✓	1	1.16	93	59	4.20	11.7	11.7	20	20	85	369	9
P015	×	4	1.45	81	244	17.91	51.6	51.6	19	23	92	–	–
P016	✓	1	1.09	94	48	3.46	10.8	10.8	25	25	70	230	15
P017	✓	1	2.44	95	86	6.41	18.8	18.7	35	35	122	733	34
P018	✓	1	2.03	94	85	6.97	17.5	17.5	25	25	115	733	13
P019	✓	1	0.53	92	26	1.98	5.5	5.5	16	16	44	169	8
P020	✓	4	12.82	96	393	33.42	77.3	77.2	131	137	162	2,025	13
P021	✓	2	6.79	97	167	14.37	37.4	37.4	82	78	119	539	30
P023	×	4	2.42	84	354	26.04	96.5	83.1	28	36	126	–	–
P025	×	4	15.80	95	403	37.46	88.9	88.8	149	155	187	–	–
P026	×	4	5.21	92	354	27.38	81.0	81.0	57	61	127	–	–
P027	×	4	0.19	85	18	1.29	4.4	4.4	15	16	15	–	–
P028	×	4	6.23	94	272	22.09	402.0	402.0	68	73	148	–	–
P030	×	4	4.08	88	461	33.49	111.8	111.8	47	55	131	–	–
P031	✓	1	2.24	95	87	7.18	20.0	20.0	31	31	115	628	29
P032	✓	3	3.39	91	252	19.29	55.1	55.1	39	43	114	1,122	30

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
P033	×	4	3.70	89	407	29.60	93.4	93.4	43	51	125	–	–
P037	×	4	15.48	96	492	41.43	100.6	100.6	126	124	185	–	–
P039	×	4	1.11	84	103	7.60	25.5	25.5	24	27	66	–	–
P040	×	4	8.76	94	433	34.02	97.1	97.0	72	80	168	–	–
P043	✓	1	1.30	93	79	6.10	18.0	18.0	18	18	97	499	32
P044	×	4	11.42	95	479	37.59	101.2	101.2	102	110	190	–	–
P045	×	4	11.32	96	324	28.23	71.1	71.0	102	106	155	–	–
P046	×	4	3.39	91	227	17.98	46.6	46.6	35	39	134	–	–
P047	×	4	4.52	92	311	24.03	69.1	69.1	55	52	119	–	–
P048	×	4	0.96	71	240	17.63	51.3	51.2	12	17	90	–	–
P049	×	4	0.83	84	100	6.84	20.8	20.8	20	24	50	–	–
P050	×	4	6.07	92	368	28.82	78.0	78.0	67	67	166	–	–
P051	✓	2	5.45	96	149	13.01	31.6	31.6	72	64	150	901	33
P052	✓	1	1.53	93	86	6.69	31.4	19.3	18	19	105	556	38
P053	×	4	0.82	73	245	16.79	50.8	50.8	16	16	91	–	–
P054	×	4	0.32	83	34	2.43	8.3	8.3	17	21	24	–	–
P055	×	4	10.41	94	455	36.66	108.7	108.7	101	109	174	–	–
P056	×	4	0.42	77	67	4.66	14.8	14.8	20	24	32	–	–
P059	×	4	1.50	89	112	8.03	24.7	24.7	42	46	54	–	–
P065	✓	1	3.81	96	118	9.53	25.2	25.2	36	36	142	510	31
P067	×	4	0.62	78	77	5.62	16.7	16.7	21	25	51	–	–
P069	×	4	7.69	94	397	30.65	78.3	78.2	82	84	148	–	–
P070	×	4	15.41	97	412	34.81	84.8	84.8	132	139	185	–	–
P071	✓	1	2.55	95	102	8.17	23.7	23.7	27	27	126	826	36
P072	✓	3	6.50	95	259	20.67	58.8	58.8	73	76	119	715	36
P079	×	4	2.85	89	271	20.28	201.7	201.7	37	43	97	–	–
P086	×	4	4.24	90	446	31.90	107.3	100.6	43	52	133	–	–
P094	✓	1	3.49	95	115	9.78	25.2	25.2	31	31	157	1,078	18
P095	✓	2	7.30	96	196	16.84	50.1	50.1	80	73	151	1,246	107
P096	×	4	4.96	93	227	18.56	51.7	51.7	71	73	109	–	–
P097	✓	2	7.37	97	164	14.67	36.4	36.3	75	77	155	1,198	63
P099	×	4	12.43	96	436	35.37	97.2	97.1	109	115	183	–	–
P100	×	4	6.58	93	469	34.04	92.3	92.3	56	64	143	–	–
P101	✓	1	0.52	92	21	1.71	6.3	6.3	21	15	48	323	15
P102	×	4	0.61	70	136	10.46	29.8	29.8	11	15	57	–	–
P103	✓	2	6.18	95	229	18.98	53.8	51.6	58	61	163	2,375	69
P104	×	4	2.64	91	156	13.21	33.6	33.6	42	41	110	–	–
P105	×	4	8.00	93	442	34.75	120.2	110.8	68	76	164	–	–
P106	×	4	0.64	77	74	5.71	16.1	16.1	13	17	49	–	–
P107	×	4	6.66	95	250	20.83	56.0	55.9	73	78	138	–	–
P108	×	4	1.54	81	193	15.57	80.8	80.8	22	26	131	–	–
P117	✓	2	7.44	97	154	13.50	45.3	45.2	87	89	130	990	54
Open (0/32 solved)													

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
P003	×	4	2.48	91	121	10.68	159.9	159.8	66	65	62	–	–
P006	×	4	1.01	84	106	7.53	23.9	23.9	30	34	51	–	–
P007	×	4	0.36	82	48	3.29	11.2	11.2	17	21	27	–	–
P009	×	4	0.19	82	20	1.48	5.0	5.0	12	15	16	–	–
P010	×	4	9.84	96	241	20.65	51.3	51.3	110	109	203	–	–
P024	×	4	0.45	80	50	3.89	43.1	43.1	19	22	30	–	–
P038	×	4	0.98	89	48	4.61	161.2	161.2	35	39	41	–	–
P042	×	4	0.14	86	10	0.75	2.8	2.8	11	14	12	–	–
P057	×	4	10.41	95	387	32.51	455.8	455.7	128	133	183	–	–
P058	×	4	7.76	95	263	22.72	434.8	434.8	116	120	130	–	–
P060	×	4	0.30	79	31	2.40	7.2	7.2	15	19	25	–	–
P061	×	4	0.40	81	44	3.23	10.6	10.6	19	23	29	–	–
P062	×	4	0.32	80	37	2.69	8.7	8.7	17	21	24	–	–
P063	×	4	4.93	92	317	25.02	328.9	328.9	58	62	129	–	–
P064	×	4	0.22	79	26	1.91	6.6	6.6	12	16	20	–	–
P066	×	4	11.15	95	438	34.79	557.0	557.0	119	126	146	–	–
P068	×	4	2.45	91	164	12.93	287.9	287.8	49	53	105	–	–
P073	×	4	0.19	84	18	1.33	5.3	5.3	12	16	16	–	–
P074	×	4	0.16	88	10	0.78	2.8	2.8	12	16	12	–	–
P075	×	4	0.34	83	49	3.26	11.3	11.3	16	20	28	–	–
P076	×	4	5.13	95	194	16.18	684.2	684.2	89	93	109	–	–
P077	×	4	0.15	86	11	0.86	2.9	2.9	10	14	12	–	–
P078	×	4	7.33	95	270	22.08	830.8	830.7	114	118	113	–	–
P080	×	4	0.15	84	14	1.06	4.0	4.0	10	14	15	–	–
P081	×	4	0.21	87	14	1.08	3.6	3.6	14	18	16	–	–
P082	×	4	1.78	91	107	8.38	104.1	104.1	35	39	90	–	–
P085	×	4	0.21	81	29	1.98	7.2	7.2	12	15	18	–	–
P112	×	4	0.33	76	63	4.20	14.0	14.0	14	18	31	–	–
P113	×	4	1.73	91	107	7.69	24.1	24.1	41	45	64	–	–
P114	×	4	1.96	90	157	11.49	33.6	33.6	37	41	91	–	–
P115	×	4	1.35	91	92	6.44	19.1	19.1	31	35	77	–	–
P116	×	4	0.26	80	35	2.47	8.3	8.3	14	18	21	–	–
Total	42/117	352	375.71		17,864	1,413.72	8,174	8,125	4,806	5,085			

Table 6: Measured quantities of Claude Fable 5.1 with web search on. The last three columns measure the use of the web. Web searches are the calls of the search tool. Pages fetched are the distinct web addresses in the agent's shell and Python commands, that is, pages downloaded and repositories cloned. Code addresses fetched are those among them that end in `.lean` or point into a GitHub repository; a solve with a positive count may rest on a published formalization.

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
Calibration (21/21 solved)																
P013	✓	1	1.88	96	53	4.09	12.3	12.3	32	32	80	332	9	0	0	0
P014	✓	1	0.23	92	9	0.73	2.7	2.7	12	12	26	131	10	0	0	0
P022	✓	1	0.44	86	40	2.84	8.1	8.1	11	11	61	160	7	0	0	0
P029	✓	1	0.20	90	11	0.82	2.7	2.7	10	10	27	100	9	0	0	0
P034	✓	1	0.12	89	7	0.52	2.1	2.1	7	7	21	50	10	0	0	0
P035	✓	1	0.14	89	8	0.62	2.5	2.5	8	8	24	63	11	0	0	0
P036	✓	1	0.24	89	13	1.03	3.6	3.6	11	11	32	101	10	0	0	0
P041	✓	1	0.10	88	7	0.54	2.0	2.0	6	6	20	159	6	0	0	0
P083	✓	1	0.66	91	50	3.87	12.7	12.6	14	14	67	249	19	0	0	0
P084	✓	1	1.74	96	12	1.82	7.8	7.7	40	40	70	3,028	16	1	3	1
P087	✓	1	1.36	92	74	5.93	19.5	19.5	20	20	91	612	64	0	0	0
P088	✓	1	0.38	93	16	1.20	5.0	5.0	18	15	33	97	14	0	0	0
P089	✓	1	0.35	94	12	0.93	4.8	4.7	17	17	29	157	25	0	0	0
P090	✓	1	0.25	93	8	0.66	3.7	3.7	13	13	25	41	15	0	0	0
P091	✓	1	0.55	92	32	2.29	9.1	9.1	18	15	52	149	22	0	0	0
P092	✓	1	0.12	88	3	0.37	4.0	4.0	7	7	21	363	30	0	0	0
P093	✓	1	2.66	95	42	4.31	9.0	9.0	40	33	135	295	7	5	0	0
P098	✓	1	0.13	91	3	0.33	2.0	2.0	9	9	19	32	14	0	0	0
P109	✓	1	0.07	88	4	0.33	3.3	3.3	7	5	16	52	17	0	0	0
P110	✓	1	0.09	92	4	0.32	2.7	2.7	9	7	14	24	15	0	0	0
P111	✓	1	0.09	93	3	0.25	1.6	1.6	7	7	14	33	10	0	0	0
Literature (30/64 solved)																
P001	×	1	1.89	92	31	4.58	15.2	15.2	33	33	85	–	–	2	15	15
P002	×	4	2.16	93	92	7.04	19.9	19.9	49	53	73	–	–	0	0	0
P004	✓	1	2.05	96	33	3.18	24.8	24.8	36	36	89	24,096	305	2	9	8
P005	✓	4	24.44	98	175	20.44	62.2	62.1	179	176	326	49,686	122	6	26	20
P008	×	2	9.97	96	157	17.34	78.8	78.7	104	105	189	–	–	5	30	25
P011	✓	1	0.73	93	6	1.16	3.0	3.0	22	18	61	1,036	15	3	4	4
P012	✓	1	2.78	96	62	5.11	13.8	13.8	36	34	114	434	21	2	7	0
P015	✓	4	11.94	95	364	30.29	77.3	77.2	124	119	208	1,352	39	2	6	0
P016	✓	1	1.83	95	54	4.97	12.0	12.0	27	27	90	215	15	0	0	0
P017	✓	1	2.59	95	96	8.14	19.3	19.2	33	33	134	739	13	0	0	0
P018	×	4	30.71	98	220	28.73	95.3	95.2	298	302	199	–	–	1	4	1
P019	✓	1	0.48	90	29	2.14	7.2	7.2	20	16	46	168	21	0	0	0
P020	✓	2	6.61	96	172	15.50	34.8	34.7	77	79	170	1,290	19	2	3	0
P021	✓	1	8.45	98	120	11.81	27.1	27.1	69	70	174	509	36	1	4	0
P023	×	4	5.05	90	431	32.24	100.6	100.5	55	63	142	–	–	1	4	0

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
P025	×	4	19.84	96	308	33.35	119.6	119.5	179	183	202	–	–	2	11	2
P026	×	4	4.02	89	352	27.17	79.0	78.9	46	54	126	–	–	4	1	0
P027	✓	4	0.96	91	27	2.69	15.5	15.5	35	29	58	5,299	24	9	9	7
P028	×	4	17.78	97	340	32.32	256.3	256.2	159	164	190	–	–	0	3	0
P030	×	4	16.74	96	428	38.49	93.5	93.5	163	170	180	–	–	0	3	0
P031	✓	2	3.90	93	186	15.47	38.9	38.9	51	53	163	888	36	0	4	0
P032	✓	1	3.81	96	96	8.61	19.9	19.9	43	43	139	768	16	0	3	0
P033	✓	2	6.86	95	213	18.63	48.5	48.5	70	69	163	1,174	40	0	2	0
P037	×	4	13.50	95	415	36.77	105.3	105.2	129	134	171	–	–	0	8	0
P039	×	4	1.98	88	144	11.78	34.5	34.5	37	41	90	–	–	1	8	0
P040	✓	3	11.12	96	329	28.73	71.1	71.0	102	101	173	2,342	41	0	2	0
P043	✓	1	2.39	95	68	6.26	16.5	16.5	34	34	106	529	33	1	3	0
P044	×	4	15.94	96	458	38.68	99.4	99.4	143	148	169	–	–	2	4	0
P045	✓	4	28.07	97	433	42.37	142.3	142.2	227	231	203	3,479	305	2	5	0
P046	×	4	1.32	89	81	6.08	17.9	17.9	36	36	64	–	–	9	4	0
P047	×	4	15.01	96	428	36.49	93.4	93.4	138	142	163	–	–	1	2	0
P048	×	4	2.47	89	153	11.62	32.5	32.5	45	49	110	–	–	0	6	0
P049	×	4	1.65	90	82	6.50	17.9	17.9	38	42	63	–	–	1	4	0
P050	×	4	4.16	89	288	23.75	63.2	63.2	55	60	127	–	–	5	11	0
P051	✓	2	2.78	92	127	11.27	24.6	24.6	36	34	158	1,099	18	0	2	0
P052	✓	2	2.79	90	146	12.88	31.4	31.4	39	40	156	818	36	1	2	0
P053	×	4	21.21	97	348	32.05	73.4	73.3	169	174	211	–	–	1	5	0
P054	✓	1	5.74	98	58	6.02	16.2	16.2	68	68	144	7,486	44	1	4	2
P055	✓	1	5.14	96	105	10.16	22.6	22.5	51	51	170	775	14	1	2	0
P056	×	4	21.64	96	488	43.91	117.7	117.6	166	173	217	–	–	3	10	2
P059	×	4	1.81	90	113	8.34	25.4	25.4	43	47	65	–	–	7	1	0
P065	✓	1	3.63	97	101	8.28	25.9	25.9	36	37	123	471	32	0	0	0
P067	×	4	0.83	83	89	6.34	20.1	20.1	24	28	63	–	–	4	1	0
P069	×	4	17.56	96	445	40.37	132.6	132.4	170	177	201	–	–	2	8	0
P070	×	4	52.34	98	449	53.41	157.5	157.2	350	354	275	–	–	2	10	9
P071	✓	2	2.62	93	147	11.71	37.4	37.4	29	31	121	739	66	0	0	0
P072	✓	2	2.94	94	132	10.95	34.6	34.5	44	46	145	911	82	0	1	0
P079	×	4	11.09	94	527	42.68	122.5	122.3	99	107	186	–	–	3	4	0
P086	×	4	21.70	96	513	48.58	105.0	104.9	161	167	240	–	–	2	4	0
P094	✓	1	5.00	97	101	9.47	31.6	31.6	51	43	160	1,064	70	0	0	0
P095	✓	2	5.57	95	183	15.50	42.5	42.5	65	63	144	1,325	46	0	0	0
P096	×	4	2.58	90	149	11.25	41.9	41.8	45	47	81	–	–	2	4	0
P097	✓	2	6.25	95	194	17.46	49.5	49.5	62	58	158	1,403	116	1	2	0
P099	×	2	4.08	91	151	14.46	48.0	48.0	44	46	171	–	–	1	4	0
P100	×	4	10.36	95	425	34.39	96.8	96.7	87	95	168	–	–	0	4	0
P101	✓	1	0.22	89	14	1.08	3.2	3.2	12	9	33	310	7	0	0	0
P102	×	4	1.05	85	96	7.00	22.6	22.6	23	26	55	–	–	1	3	0
P103	✓	3	13.68	97	274	25.72	83.5	83.4	120	124	185	2,458	120	0	3	0

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
P104	×	4	13.37	97	262	21.65	63.5	63.4	141	145	151	–	–	0	3	0
P105	×	4	1.26	87	73	5.99	17.7	17.7	27	31	67	–	–	4	2	0
P106	×	4	8.55	94	280	23.46	73.0	72.9	87	87	180	–	–	2	4	0
P107	×	4	21.16	96	493	44.79	114.3	114.2	171	177	193	–	–	0	4	0
P108	×	4	1.28	84	97	8.30	23.0	22.9	24	28	63	–	–	2	3	0
P117	✓	2	3.33	94	144	11.68	33.2	33.2	42	44	142	1,004	35	0	0	0
Open (0/32 solved)																
P003	×	4	1.41	90	81	7.08	199.2	199.2	43	47	58	–	–	4	0	0
P006	×	4	2.19	91	111	8.49	24.5	24.4	45	49	81	–	–	3	0	0
P007	×	4	0.41	84	45	3.17	10.9	10.9	18	22	27	–	–	2	0	0
P009	×	4	0.24	81	25	1.89	6.3	6.2	12	15	23	–	–	2	0	0
P010	×	4	5.19	95	169	14.68	39.6	39.6	73	77	143	–	–	6	8	0
P024	×	4	0.31	77	32	2.53	7.8	7.8	22	15	28	–	–	8	0	0
P038	×	4	0.69	88	26	2.49	6.5	6.5	27	29	42	–	–	6	3	0
P042	×	4	0.18	87	10	0.84	2.7	2.7	11	15	18	–	–	0	0	0
P057	×	4	16.77	97	297	28.87	683.8	683.7	201	203	143	–	–	7	5	0
P058	×	4	9.93	96	234	21.25	539.6	539.5	136	138	143	–	–	9	1	0
P060	×	4	0.45	85	26	2.20	6.2	6.1	21	25	29	–	–	7	0	0
P061	×	4	2.09	92	78	6.97	241.2	241.1	59	63	61	–	–	6	2	0
P062	×	4	0.36	83	28	2.25	6.8	6.8	17	21	26	–	–	3	0	0
P063	×	4	1.07	84	118	8.28	24.2	24.2	29	33	65	–	–	7	2	0
P064	×	4	0.38	83	32	2.46	7.5	7.5	18	22	25	–	–	4	0	0
P066	×	2	21.08	97	307	30.63	1,080.1	1,080.0	196	199	178	–	–	6	9	0
P068	×	4	1.19	87	74	6.25	90.4	90.4	43	46	53	–	–	5	16	0
P073	×	4	0.39	83	18	1.79	5.0	5.0	23	21	32	–	–	10	1	0
P074	×	4	0.16	88	9	0.71	2.6	2.6	10	14	13	–	–	0	0	0
P075	×	4	0.57	85	43	3.32	11.1	11.1	24	26	33	–	–	10	0	0
P076	×	4	6.78	95	211	18.45	546.8	546.7	121	107	126	–	–	9	3	0
P077	×	4	0.20	82	10	1.00	3.1	3.1	14	14	19	–	–	6	1	0
P078	×	4	0.38	84	24	2.04	7.2	7.2	23	21	25	–	–	9	4	0
P080	×	4	0.24	84	14	1.23	4.6	4.6	14	17	19	–	–	4	1	0
P081	×	4	0.25	85	22	1.61	5.6	5.6	14	18	19	–	–	2	0	0
P082	×	4	0.60	84	52	3.88	13.6	13.6	31	28	40	–	–	12	8	0
P085	×	4	0.30	82	28	2.11	9.9	9.8	18	17	24	–	–	6	0	0
P112	×	4	1.40	91	59	4.89	15.1	15.0	51	47	66	–	–	9	20	5
P113	×	4	3.80	93	117	9.82	28.4	28.4	81	82	86	–	–	7	24	12
P114	×	4	0.71	89	40	3.14	9.7	9.7	32	35	35	–	–	5	11	0
P115	×	4	1.96	91	112	8.20	26.2	26.2	54	50	75	–	–	8	5	0
P116	×	4	1.29	91	53	4.36	13.6	13.6	43	47	43	–	–	9	9	0
Total	51/117	332	649.48		16,516	1,468.25	7,449	7,444	7,218	7,366				301		

Table 7: Measured quantities of Claude Opus 5.5 where web search is not allowed.

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
Calibration (21/21 solved)													
P013	✓	1	0.41	90	27	0.83	6.2	6.2	18	14	42	223	22
P014	✓	1	0.07	84	6	0.18	1.8	1.7	6	5	18	123	9
P022	✓	1	0.35	88	21	0.69	4.3	4.3	13	12	41	154	7
P029	✓	1	0.13	87	6	0.22	2.0	2.0	9	8	23	93	10
P034	✓	1	0.09	87	4	0.17	2.6	2.6	7	7	18	61	17
P035	✓	1	0.07	83	5	0.16	1.7	1.7	7	5	18	53	10
P036	✓	1	0.07	81	8	0.24	2.1	2.1	6	5	20	106	10
P041	✓	1	0.08	88	5	0.15	1.4	1.4	6	6	16	138	6
P083	✓	1	0.32	91	20	0.61	4.8	4.8	15	13	35	163	14
P084	✓	1	1.38	93	71	2.15	14.2	14.2	20	19	102	537	42
P087	✓	1	0.74	93	43	1.27	10.2	10.2	17	17	62	428	43
P088	✓	1	0.09	84	9	0.26	2.0	2.0	8	6	20	89	7
P089	✓	1	0.21	90	12	0.38	3.2	3.2	12	10	26	131	10
P090	✓	1	0.18	91	7	0.25	2.3	2.3	11	11	23	35	12
P091	✓	1	0.25	91	18	0.51	5.5	5.5	13	12	29	156	31
P092	✓	1	0.31	93	8	0.33	3.5	3.5	17	16	29	360	14
P093	✓	1	1.33	95	43	1.45	8.3	8.3	36	25	76	312	14
P098	✓	1	0.09	89	3	0.13	2.1	2.1	7	7	17	33	14
P109	✓	1	0.04	84	3	0.11	1.4	1.4	5	4	13	47	8
P110	✓	1	0.04	89	2	0.07	3.0	3.0	4	4	10	20	19
P111	✓	1	0.03	86	2	0.06	1.8	1.8	4	3	10	23	14
Literature (24/64 solved)													
P001	×	4	0.53	84	32	1.15	6.3	6.3	24	27	33	–	–
P002	×	4	0.30	85	19	0.66	3.9	3.9	16	20	23	–	–
P004	×	4	0.63	86	53	1.74	52.6	52.6	29	33	32	–	–
P005	×	4	0.17	83	17	0.52	3.7	3.7	10	14	17	–	–
P008	×	4	6.36	95	213	7.83	442.2	442.1	107	111	101	–	–
P011	✓	1	2.73	95	98	3.48	15.9	15.9	32	30	123	640	11
P012	✓	1	0.41	89	37	1.04	7.2	7.2	12	10	51	337	12
P015	×	4	1.01	85	145	4.11	24.5	24.5	27	28	51	–	–
P016	✓	1	0.70	93	31	1.01	6.8	6.8	23	19	57	172	14
P017	×	1	0.82	89	68	1.95	28.9	28.9	16	14	96	–	–
P018	✓	1	1.13	94	52	1.60	10.3	10.3	25	22	75	509	10
P019	✓	1	0.24	88	17	0.53	3.5	3.5	12	10	35	154	8
P020	✓	2	4.59	94	206	7.14	53.2	53.1	58	49	150	1,607	21
P021	✓	1	3.56	96	102	3.73	21.4	21.4	41	41	130	606	20
P023	×	4	0.57	67	163	4.71	29.6	29.6	20	22	79	–	–
P025	×	4	13.85	96	380	14.60	125.5	125.3	139	137	165	–	–
P026	×	4	6.49	93	384	12.46	77.3	77.2	70	74	138	–	–
P027	×	4	0.11	85	7	0.23	1.6	1.6	7	11	11	–	–
P028	×	4	4.56	93	161	6.29	203.1	203.1	69	70	109	–	–

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
P030	×	4	6.68	92	412	13.48	74.9	74.8	71	74	146	–	–
P031	✓	1	1.16	92	66	2.21	12.5	12.5	19	19	88	514	37
P032	×	4	6.38	91	469	14.79	90.4	90.1	71	75	138	–	–
P033	×	4	3.89	89	403	12.23	90.8	90.8	43	47	134	–	–
P037	×	4	6.33	94	250	9.30	111.2	111.2	73	70	157	–	–
P039	×	4	0.68	76	65	2.60	28.7	28.6	17	21	44	–	–
P040	✓	4	8.18	94	369	12.83	67.8	67.8	94	81	161	1,776	37
P043	✓	1	0.49	91	37	1.04	8.0	8.0	14	14	51	342	25
P044	✓	3	8.11	95	348	11.97	65.1	65.1	83	80	172	2,098	62
P045	×	4	4.72	90	401	12.51	72.2	72.2	48	50	134	–	–
P046	×	4	3.23	94	153	4.66	37.8	37.8	54	51	112	–	–
P047	×	4	2.81	88	285	8.89	50.3	50.3	35	36	112	–	–
P048	×	4	0.55	68	145	4.36	24.1	24.1	11	15	54	–	–
P049	×	4	0.45	80	48	1.47	8.6	8.6	16	19	31	–	–
P050	×	4	2.39	91	159	5.41	27.5	27.5	37	39	96	–	–
P051	✓	1	1.43	93	87	2.83	14.0	14.0	18	17	107	683	8
P052	✓	1	1.14	90	79	2.61	16.1	16.0	21	15	97	455	32
P053	×	4	9.09	95	363	12.52	62.6	62.6	86	89	159	–	–
P054	×	4	0.13	81	11	0.35	2.2	2.2	8	12	13	–	–
P055	✓	1	1.90	95	77	2.70	23.7	23.7	27	25	102	637	84
P056	×	4	0.27	80	35	1.01	6.1	6.1	17	19	22	–	–
P059	×	4	0.52	82	69	1.93	12.3	12.3	25	23	44	–	–
P065	✓	1	2.09	93	68	2.78	25.0	25.0	34	32	91	451	14
P067	×	4	0.56	82	50	1.59	8.9	8.9	24	27	33	–	–
P069	×	4	7.04	93	418	13.39	87.9	87.8	73	77	135	–	–
P070	×	4	11.67	96	411	14.46	68.8	68.8	104	104	182	–	–
P071	✓	1	3.40	96	105	3.79	19.9	19.9	35	32	132	741	18
P072	✓	2	3.05	93	174	5.66	32.5	32.4	37	36	108	579	52
P079	×	4	0.36	75	55	1.72	35.0	35.0	16	19	34	–	–
P086	×	4	8.15	94	396	13.23	97.7	97.7	91	95	126	–	–
P094	✓	1	2.93	96	83	2.80	17.0	17.0	35	35	122	1,118	22
P095	✓	1	3.53	96	113	3.69	28.8	28.8	39	35	158	1,070	84
P096	×	4	4.09	93	178	6.44	75.8	75.8	49	50	145	–	–
P097	✓	2	2.58	93	131	4.58	23.3	23.2	26	28	155	1,123	38
P099	×	2	2.85	92	209	6.56	41.5	41.5	30	31	134	–	–
P100	✓	3	7.58	94	334	11.50	63.0	63.0	72	74	170	2,184	53
P101	✓	1	0.16	86	13	0.39	2.9	2.9	11	7	28	320	7
P102	×	4	0.24	67	68	1.79	12.9	12.9	7	10	36	–	–
P103	✓	2	6.36	95	220	8.05	54.5	54.5	59	59	159	2,278	159
P104	×	4	3.18	91	161	5.17	27.6	27.6	52	48	113	–	–
P105	×	4	0.47	79	35	1.27	7.5	7.5	13	17	36	–	–
P106	×	4	2.46	91	109	3.68	20.4	20.4	33	36	113	–	–
P107	×	4	0.44	81	56	1.60	10.0	10.0	19	20	37	–	–

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)
P108	×	4	0.42	77	28	1.10	6.8	6.8	11	15	35	–	–
P117	✓	2	2.27	92	148	4.89	24.2	24.2	35	34	138	1,073	7
Open (0/32 solved)													
P003	×	4	0.36	87	24	0.86	30.6	30.6	20	24	23	–	–
P006	×	4	0.67	84	46	1.57	9.8	9.8	27	30	42	–	–
P007	×	4	0.23	84	23	0.68	5.1	5.1	14	18	20	–	–
P009	×	4	0.17	85	10	0.36	2.7	2.7	12	15	16	–	–
P010	×	4	0.72	88	59	1.74	13.9	13.9	27	30	47	–	–
P024	×	4	0.15	85	10	0.34	2.4	2.4	10	14	14	–	–
P038	×	4	0.23	86	13	0.46	2.9	2.9	15	18	20	–	–
P042	×	4	0.13	88	6	0.22	21.3	21.3	10	14	11	–	–
P057	×	4	4.37	95	132	5.10	272.4	272.3	114	115	66	–	–
P058	×	4	8.52	96	196	7.89	411.6	411.6	154	157	97	–	–
P060	×	4	0.21	85	14	0.48	3.4	3.4	15	18	19	–	–
P061	×	4	0.27	79	25	0.83	5.8	5.8	18	19	24	–	–
P062	×	4	0.23	85	15	0.51	3.6	3.6	15	19	18	–	–
P063	×	4	0.24	77	36	1.02	6.7	6.7	13	16	30	–	–
P064	×	4	0.18	84	12	0.42	3.8	3.7	12	15	15	–	–
P066	×	4	7.03	96	237	8.54	489.4	489.4	124	125	108	–	–
P068	×	4	0.62	85	42	1.63	65.7	65.7	27	29	41	–	–
P073	×	4	0.14	87	9	0.29	2.6	2.6	10	14	12	–	–
P074	×	4	0.20	89	11	0.37	4.1	4.1	14	18	15	–	–
P075	×	4	0.21	85	18	0.56	3.9	3.9	13	17	18	–	–
P076	×	4	0.75	87	56	1.97	117.2	117.2	31	35	35	–	–
P077	×	4	0.13	85	7	0.26	2.7	2.7	9	13	13	–	–
P078	×	4	0.16	85	12	0.38	2.9	2.8	12	14	16	–	–
P080	×	4	0.13	86	8	0.26	2.0	2.0	10	13	11	–	–
P081	×	4	0.16	88	10	0.32	3.1	3.1	12	16	12	–	–
P082	×	4	0.40	81	39	1.36	58.6	58.6	18	22	35	–	–
P085	×	4	0.14	86	11	0.34	3.7	3.7	9	13	13	–	–
P112	×	4	0.27	79	27	0.85	5.2	5.2	19	18	30	–	–
P113	×	4	0.50	88	33	1.06	9.3	9.2	24	27	33	–	–
P114	×	4	0.90	90	51	1.63	13.7	13.7	36	35	50	–	–
P115	×	4	0.47	85	49	1.40	10.2	10.2	20	22	42	–	–
P116	×	4	0.24	85	20	0.63	6.5	6.5	14	18	19	–	–
Total	45/117	340	230.58		11,656	391.23	4,498	4,495	3,649	3,734			

Table 8: Measured quantities of Claude Opus 5.5 with web search on. The last three columns measure the use of the web. Web searches are the calls of the search tool. Pages fetched are the distinct web addresses in the agent's shell and Python commands, that is, pages downloaded and repositories cloned. Code addresses fetched are those among them that end in `.lean` or point into a GitHub repository; a solve with a positive count may rest on a published formalization.

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
Calibration (21/21 solved)																
P013	✓	1	0.49	90	33	0.99	7.6	7.6	16	14	49	253	25	0	0	0
P014	✓	1	0.13	90	6	0.22	2.0	2.0	8	8	21	134	9	0	0	0
P022	✓	1	0.40	91	14	0.54	3.3	3.3	15	14	38	142	7	0	0	0
P029	✓	1	0.13	86	8	0.27	2.2	2.2	8	7	25	97	10	0	0	0
P034	✓	1	0.07	85	4	0.15	1.6	1.6	5	5	18	49	10	0	0	0
P035	✓	1	0.07	85	5	0.17	1.7	1.7	6	5	19	46	11	0	0	0
P036	✓	1	0.10	86	7	0.23	2.1	2.1	7	6	21	90	10	0	0	0
P041	✓	1	0.06	83	4	0.15	1.4	1.4	5	4	18	141	7	0	0	0
P083	✓	1	0.28	89	22	0.65	5.7	5.7	12	10	39	206	18	0	0	0
P084	✓	1	0.77	94	11	0.58	5.3	5.3	25	23	51	3,022	21	1	2	0
P087	✓	1	1.28	95	56	1.69	9.8	9.8	24	24	76	458	11	0	0	0
P088	✓	1	0.08	90	5	0.16	1.6	1.6	8	6	16	90	7	0	0	0
P089	✓	1	0.12	91	3	0.14	1.8	1.8	8	8	19	193	11	0	0	0
P090	✓	1	0.25	94	5	0.23	2.2	2.2	15	15	24	37	11	0	0	0
P091	✓	1	0.42	93	19	0.60	4.5	4.5	17	16	37	153	8	0	0	0
P092	✓	1	0.09	86	2	0.12	1.5	1.5	6	6	20	362	12	0	0	0
P093	✓	1	0.89	94	30	1.05	5.9	5.9	21	20	63	217	8	0	0	0
P098	✓	1	0.14	92	3	0.13	1.6	1.6	9	9	19	29	11	0	0	0
P109	✓	1	0.06	89	4	0.12	1.2	1.2	6	5	14	50	7	0	0	0
P110	✓	1	0.06	90	4	0.11	1.4	1.4	5	5	14	33	7	0	0	0
P111	✓	1	0.04	91	2	0.07	0.9	0.9	4	4	12	29	6	0	0	0
Literature (34/64 solved)																
P001	×	1	1.68	96	22	1.10	13.5	13.5	40	33	76	–	–	4	10	10
P002	×	4	0.30	84	19	0.66	3.7	3.7	17	19	24	–	–	0	0	0
P004	✓	1	2.01	96	19	1.16	24.6	24.6	40	39	85	32,495	310	3	12	12
P005	✓	4	20.54	97	127	11.05	71.2	71.1	186	165	231	49,485	124	9	7	4
P008	×	1	1.66	95	26	1.26	7.3	7.3	37	33	85	–	–	2	11	10
P011	✓	1	1.55	94	64	2.03	10.8	10.8	25	23	100	626	10	1	4	1
P012	✓	1	0.52	89	50	1.39	8.5	8.5	11	11	65	410	11	0	0	0
P015	✓	2	6.76	95	231	8.61	39.4	39.3	71	72	169	1,252	29	2	2	0
P016	✓	1	0.59	92	26	0.84	6.0	6.0	18	17	52	136	15	0	0	0
P017	✓	1	1.24	92	68	2.06	11.5	11.4	21	21	94	704	21	0	0	0
P018	✓	1	1.41	93	69	2.11	11.7	11.7	27	24	102	655	12	1	0	0
P019	✓	1	0.15	80	13	0.43	2.9	2.9	9	7	30	143	8	0	0	0
P020	✓	2	4.17	93	169	6.24	28.2	28.2	57	56	159	1,356	23	3	1	0
P021	✓	1	2.14	94	90	3.25	15.5	15.4	29	29	118	590	18	1	1	0
P023	×	4	7.50	93	499	15.45	89.1	89.1	77	82	147	–	–	4	6	0

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
P025	×	4	17.70	96	404	17.13	101.6	101.5	176	163	176	–	–	3	7	2
P026	×	4	4.94	92	425	12.75	79.0	78.9	52	57	118	–	–	2	0	0
P027	×	4	0.15	83	7	0.29	1.8	1.8	9	12	15	–	–	4	0	0
P028	×	4	8.19	94	252	10.14	115.4	115.3	122	102	138	–	–	0	2	0
P030	✓	3	6.76	93	304	10.74	56.5	56.5	83	76	159	1,712	50	2	5	0
P031	✓	1	1.79	95	58	1.96	11.9	11.9	33	31	100	562	17	0	3	0
P032	✓	1	1.83	94	80	2.45	14.2	14.2	29	28	109	753	14	0	2	0
P033	✓	2	3.94	93	180	6.47	30.9	30.9	52	51	146	1,130	24	1	2	0
P037	×	4	13.88	96	416	14.07	73.3	73.2	141	134	193	–	–	0	4	0
P039	×	4	20.35	96	446	17.30	106.9	106.8	169	164	201	–	–	2	4	0
P040	✓	2	5.03	95	194	6.17	34.3	34.2	57	59	160	1,674	33	0	2	0
P043	✓	1	0.94	94	39	1.23	9.5	9.5	28	25	64	410	31	0	3	0
P044	✓	3	10.80	96	276	10.68	48.5	48.4	110	109	171	1,475	27	0	0	0
P045	×	4	20.65	96	428	18.49	82.7	82.6	167	166	216	–	–	4	5	0
P046	×	4	1.12	88	68	2.24	13.0	13.0	35	35	48	–	–	6	4	0
P047	×	4	18.49	95	501	20.12	99.2	99.2	165	163	195	–	–	3	7	4
P048	×	4	0.69	80	102	2.85	17.2	17.2	22	21	56	–	–	1	3	0
P049	×	4	1.12	89	49	1.77	9.0	9.0	33	34	55	–	–	2	2	0
P050	×	3	9.41	95	311	11.08	58.0	57.9	98	92	157	–	–	2	2	0
P051	✓	2	4.74	95	124	5.14	21.4	21.4	46	46	174	984	10	1	1	0
P052	✓	1	9.77	98	66	4.83	85.2	85.1	102	91	189	2,913	44	1	5	4
P053	×	4	19.20	96	515	19.97	89.8	89.8	156	155	193	–	–	1	4	0
P054	✓	1	3.82	97	54	2.39	14.5	14.5	56	56	120	7,418	48	1	4	1
P055	✓	1	1.33	93	78	2.27	13.9	13.9	21	20	102	621	29	0	0	0
P056	✓	2	12.65	97	189	7.97	36.0	36.0	111	107	239	7,523	40	2	7	4
P059	×	4	0.99	87	59	1.99	11.3	11.3	37	35	47	–	–	10	1	0
P065	✓	1	1.42	94	65	1.97	11.7	11.7	23	21	90	475	14	0	0	0
P067	×	4	0.54	83	44	1.43	8.3	8.3	21	25	41	–	–	2	0	0
P069	×	4	12.63	95	501	16.57	93.4	93.3	124	127	169	–	–	4	3	0
P070	×	4	32.15	98	359	16.98	105.9	105.8	290	243	254	–	–	1	3	2
P071	✓	1	1.35	92	79	2.66	14.8	14.8	22	17	107	724	31	0	0	0
P072	✓	1	1.49	93	76	2.29	12.7	12.7	26	24	107	666	15	2	1	0
P079	✓	3	7.40	93	294	10.94	56.9	56.8	80	76	162	1,562	89	3	2	0
P086	×	4	17.81	95	494	19.94	90.0	89.9	155	146	227	–	–	2	3	0
P094	✓	1	2.01	93	84	3.09	17.2	17.2	23	22	128	1,146	54	0	0	0
P095	✓	2	5.07	95	161	6.03	31.0	31.0	59	54	150	1,221	36	0	0	0
P096	×	4	6.45	94	168	7.48	66.2	66.1	98	78	157	–	–	1	2	0
P097	✓	2	3.27	93	132	4.98	22.9	22.9	38	34	166	1,140	27	1	1	0
P099	×	1	1.86	95	75	2.59	68.1	68.1	27	26	97	–	–	1	3	0
P100	✓	3	9.12	94	365	13.17	61.7	61.6	89	89	172	2,442	28	1	1	0
P101	✓	1	0.23	91	10	0.34	2.2	2.2	10	10	28	299	7	0	0	0
P102	×	4	0.60	82	55	1.74	9.8	9.7	18	21	45	–	–	1	3	0
P103	✓	2	4.21	93	181	6.74	36.0	36.0	48	45	170	1,713	78	0	2	0

continued

Problem	Verdict	Sessions used	Input tokens (M)	of which cached (%)	Output tokens (k)	Cost (USD)	Total time (min)	Working time (min)	Tool calls	Model calls	Largest prompt (k tokens)	Proof length (lines)	Compile time (s)	Web searches	Pages fetched	Code addresses fetched
P104	×	4	4.52	93	150	5.38	27.2	27.2	64	66	125	–	–	1	2	0
P105	×	4	0.88	84	63	2.12	12.2	12.2	23	24	53	–	–	4	3	0
P106	×	4	6.48	94	206	7.20	40.8	40.8	84	77	144	–	–	2	3	0
P107	×	4	9.20	95	291	10.87	52.3	52.2	93	95	173	–	–	0	4	0
P108	×	4	0.92	87	34	1.42	6.8	6.8	27	28	40	–	–	2	3	0
P117	✓	1	1.83	93	100	3.30	16.1	16.1	21	20	126	1,157	7	0	0	0
Open (0/32 solved)																
P003	×	4	0.47	91	19	0.67	5.0	5.0	25	27	27	–	–	0	0	0
P006	×	4	0.61	81	50	1.66	9.2	9.2	23	24	54	–	–	3	0	0
P007	×	4	0.71	87	33	1.26	7.7	7.7	30	29	49	–	–	10	3	0
P009	×	4	0.47	90	25	0.83	5.6	5.6	26	28	23	–	–	0	0	0
P010	×	4	1.05	91	57	1.82	12.6	12.5	37	37	50	–	–	3	0	0
P024	×	4	0.64	91	28	0.95	6.7	6.6	27	28	42	–	–	4	1	1
P038	×	4	0.33	86	15	0.60	3.8	3.8	18	21	23	–	–	3	0	0
P042	×	4	0.13	88	6	0.22	2.0	2.0	8	12	12	–	–	0	0	0
P057	×	4	4.99	95	97	4.66	102.2	102.1	101	99	97	–	–	10	5	2
P058	×	4	12.59	96	240	10.67	578.9	578.8	177	173	141	–	–	13	6	0
P060	×	4	0.21	89	11	0.37	2.7	2.7	13	17	15	–	–	0	0	0
P061	×	4	0.41	85	24	0.84	5.0	5.0	23	24	29	–	–	3	0	0
P062	×	4	0.27	84	14	0.54	3.4	3.4	15	18	21	–	–	3	0	0
P063	×	4	0.44	80	36	1.23	7.2	7.2	23	22	37	–	–	9	3	0
P064	×	4	0.40	88	17	0.66	4.8	4.8	21	24	29	–	–	2	0	0
P066	×	4	13.26	97	239	10.83	636.8	636.7	176	175	134	–	–	8	9	0
P068	×	4	0.36	84	20	0.73	4.5	4.5	20	21	27	–	–	6	0	0
P073	×	4	0.21	86	9	0.37	3.3	3.3	13	16	17	–	–	3	1	0
P074	×	4	0.13	89	6	0.21	1.6	1.6	8	12	11	–	–	0	0	0
P075	×	4	0.26	82	17	0.62	3.8	3.8	16	17	23	–	–	8	0	0
P076	×	4	0.29	84	13	0.54	4.5	4.5	18	18	27	–	–	4	4	0
P077	×	4	0.17	85	8	0.31	2.4	2.4	9	13	17	–	–	1	0	0
P078	×	4	0.21	84	13	0.47	3.1	3.1	14	15	19	–	–	4	1	0
P080	×	4	0.16	87	9	0.32	4.0	4.0	10	14	14	–	–	0	0	0
P081	×	4	0.18	88	10	0.34	3.7	3.7	11	15	14	–	–	0	0	0
P082	×	4	0.32	84	16	0.63	3.8	3.8	21	20	25	–	–	9	1	0
P085	×	4	0.13	84	10	0.32	3.8	3.8	8	11	16	–	–	1	0	0
P112	×	4	1.46	91	44	1.77	9.6	9.5	51	47	76	–	–	10	21	12
P113	×	4	2.30	93	78	2.83	16.2	16.2	77	65	72	–	–	10	14	10
P114	×	4	1.41	92	34	1.49	8.5	8.5	43	42	68	–	–	9	5	1
P115	×	4	0.80	88	50	1.63	12.2	12.2	34	33	43	–	–	10	1	0
P116	×	4	0.25	84	16	0.56	4.0	4.0	13	17	22	–	–	3	0	0
Total	55/117	313	435.48		12,613	478.68	3,991	3,988	5,577	5,379				256		